Model Driven Development for Business Performance Management

P. Chowdhary
K. Bhaskaran
N. S. Caswell
H. Chang
T. Chao
S.-K. Chen
M. Dikun
H. Lei
J.-J. Jeng
S. Kapoor
C. A. Lang
G. Mihaila
I. Stanoi
L. Zeng

Business process integration and monitoring provides an invaluable means for an enterprise to adapt to changing conditions. However, developing such applications using traditional methods is challenging because of the intrinsic complexity of integrating large-scale business processes and existing applications. Model Driven Development™ (MDD™) is an approach to developing applications—from domain-specific models to platform-sensitive models—that bridges the gap between business processes and information technology. We describe the MDD framework and methodology used to create the IBM Business Performance Management (BPM) solution. We describe how we apply model-driven techniques to BPM and present a scenario from a pilot project in which these techniques were applied. Technical details on models and transformation are presented. Our framework uses and extends the IBM business observation metamodel and introduces a data warehouse metamodel and other platform-specific and transformational models. We discuss our lessons learned and present the general guidelines for using MDD to develop enterprise-scale applications.

INTRODUCTION

Business Performance Management (BPM)[1–4] has emerged as a critical discipline to enable enterprises to manage their business solutions in an on demand fashion. Gartner has coined the term *business activity monitoring* (BAM)[3] and predicts significant growth in this area. With such wide interest, the market has been flooded with terminology similar to BAM, creating some confusion. It is not the intent of this paper to attempt to clarify this confusion; we direct the reader to Reference 2, which describes in detail the IBM BPM approach and how it is positioned with respect to competing terminology.

Even before Gartner drew attention to the need large enterprises had for BAM, Stephan Haeckel of the IBM Advanced Business Institute described in his book the transformation from a *make-and-sell* organization to a *sense-and-respond* organization.[5] Inspired by Haeckel's work and market needs, various IBM divisions have been developing meth-

odologies, frameworks, tools, and software components to support adaptive enterprises. IBM Research has been active in the development of this technology and has sponsored several pilot projects[6,7] to better understand its applicability and benefits.

From the IBM point of view,[1,2] a BPM system is an on demand platform for business performance monitoring and control. It takes data monitored from targeted business solutions and events, invokes BPM services, and renders actions back to the target business solutions. The BPM reference architecture and its components are described in Reference 2, p. 37. The BPM architecture for our solution closely follows the reference architecture.

Originally, models were used in software development solely for the purpose of documentation and presentation. The advent of extensible tools[8,9] brought about Model Driven Development** (MDD**). With it, users could create new notations to express an artifact in a model and attach software components to it. This ability makes it possible to automate the transformation of user-annotated enhanced models into deployed code and services. In recent years, new emphasis in research and development has focused on MDD[10,11] as an alternative to traditional software development methodology.

Once BPM systems are implemented, they are very hard to change because they are engineered as software development solutions that are linear and rigid or because the monitoring solution derives from process models. Solutions derived from processes are flexible but not comprehensive enough to include the nonprocess metrics needed to represent the full state of a business. Thus, a BPM approach not based on models can fall short of fully meeting business needs.

The abstraction of the BPM solution to higher-level models, as we propose, overcomes the shortcomings of BPM alone. It enables business analysts and system architects to contribute directly to the solution. The MDD approach to BPM means that business goals can be defined independent of an information technology (IT) platform. Business-level models either provide linkage to or can be automatically transformed directly to IT-level models using transformation routines. MDD can quickly reflect changing business goals and monitoring needs through models. This paper explains our modeling approach to BPM and demonstrates the ease of use of our modeling framework. We also describe the modeling annotations of various artifacts that make up the BPM solution and the process of automating the production of code from model to deployment.

OVERVIEW OF OUR APPROACH

We have developed a technology framework and software platform to represent a BPM solution by using formal BPM models in a top-down fashion. We have also developed model software components that can be attached to a modeling tool and that can automate the transformation of BPM models into deployable code.

The BPM modeling framework is a refinement and augmentation of the observation metamodel. At a high level, it captures the following aspects of a BPM solution: information gathering from real-time business events and other data sources, information aggregation to calculate business metrics, recognition of situations warranting business actions, and the invocation of actions that address the situations detected.

To enable the representation of a solution using models, we decompose various aspects of BPM into smaller manageable components, called BPM *elements*. These elements, together with their operational semantics, comprise the BPM *metamodel*. The elements are designed with ease of use in mind and are at the same time rich enough to represent a complete BPM solution. To represent BPM elements, we chose Unified Modeling Language** (UML**) with UML Version 2.0 (UML2) profiles[12] for extension. We selected IBM Rational* Software Architect (RSA), which supports UML model extensions. ***Figure 1*** represents the various BPM elements and their relationships with one another. Together, they collectively comprise the BPM metamodel.

The UML representation of the metamodel helps if one is designing a solution from the beginning, but if someone has an existing solution in some representation and does not want to start with BPM UML models, we also provide a representation of the BPM metamodel as an Extensible Markup Language (XML) definition.[13] This enables users to transform their solution into an XML representation.

A BPM solution as a UML model can be created with the elements shown in Figure 1 by using the RSA modeling tool. One can then use the software component plug-in to perform an automated trans-

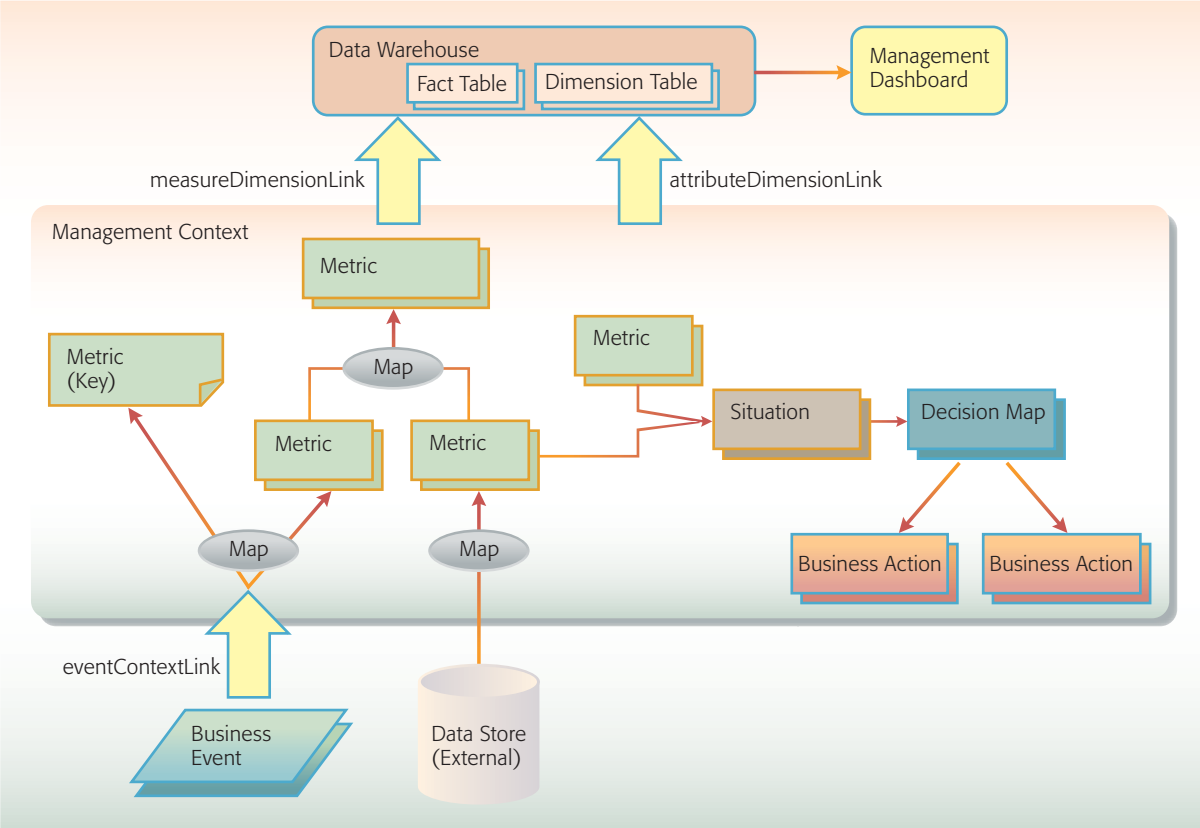

Figure 1
BPM metamodel: elements and operational semantics

formation of the model into a deployment module (e.g., a monitor runtime, data warehouse, or dashboard module). The automated transformation hides the complex inner workings of the transformations that create intermediate metamodels. The code is generated based on these intermediate models and finally packaged for deployment. One can make changes to the intermediate models to further augment the model if desired, but normally it is not needed. One also has the choice to go back to the UML model and make changes as needed. This can be an iterative process until a satisfactory version of the model is created. With MDD, business analysts can visually design BPM solutions without development team involvement.

Figure 2 shows the BPM tooling flow and user roles. In the modeling stage, one can start with either an XML editor or RSA. Both approaches can generate an observation model (OM), represented in the XML Metadata Interchange (XMI**) format.[14] In this paper, we focus on using the RSA approach.

Once the model is created, the transformation generates intermediate models, such as the OM and the data warehouse model. The intermediate models then generate code. In the figure, we show observation, action, and visibility code being generated. The code generated then generates what we call a *deployment module*, and each module contains multiple services.

The next step is to deploy these service components to their respective runtime environments. ***Figure 3*** shows input sources and the deployment of BPM components, including their respective software. (The indicated execution steps are discussed later in the section on sample scenario execution.)

Input sources are modeled in UML. (Later we describe how to represent such input sources.) If an ad hoc event is input, then the data is sent through the event infrastructure, which could be, for example, an Enterprise Service Bus (ESB)[15] or Common Event Infrastructure (CEI)[16] (our choice).

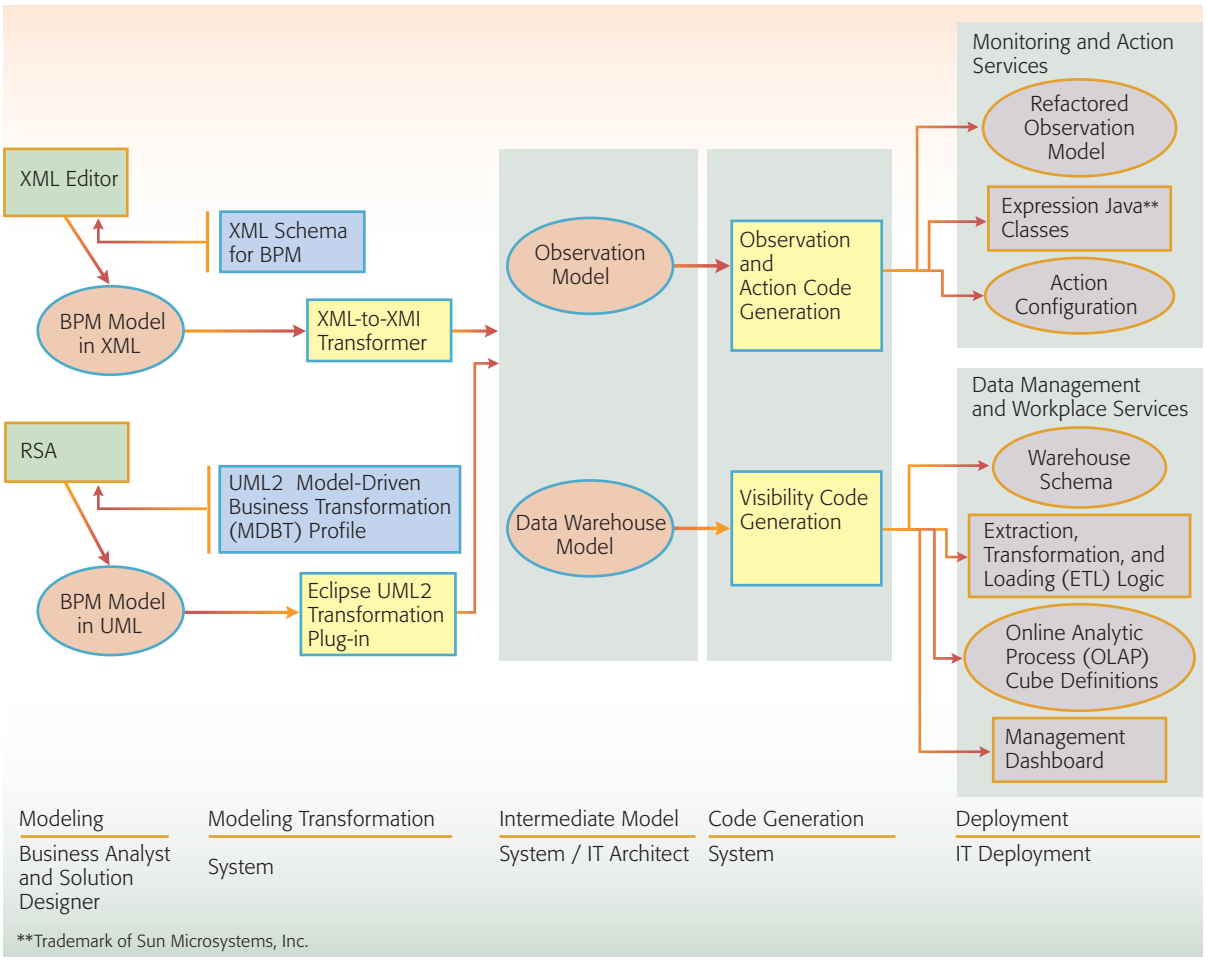

Figure 2
BPM tooling flow

BPM services is a collection of runtime services and their deployment scripts generated from the BPM models. The BPM services collectively process incoming data, correlate and compute metrics, evaluate business or IT situations, send alert notifications through preferred channels, and store processed data in an operational data store (ODS). The Extraction, Transformation, and Loading (ETL) service processes data by pulling it from the ODS on a periodic basis, transforming the data, and storing transformed data in the data warehouse. The management dashboard retrieves data from the data warehouse and generates reports.

Advantages of the MDD approach

Our MDD approach to BPM has advantages over general IT systems development because the expressive power of our BPM metamodel is purposely restricted to generic and relatively simple constructs, such as metrics, maps, dimensions, business events, situations, and actions (Figure 1). By restricting the expressive power, we assure that a well-defined, nonambiguous solution is generated. In addition, our model takes a holistic view of monitoring requirements and can represent them with formal models. Our solution also performs basic model validation to assure that the BPM elements used are semantically correct and can be automatically transformed into deployable code. Our solution is deterministic and repetitive and supports the iterative MDD approach. It also generates a default dashboard component that can be deployed on the IBM DB2* Alphablox[17]; hence, one can view the output of a modeled solution and change or add features to the models.

This basic approach has been further refined and the software transformation components made more

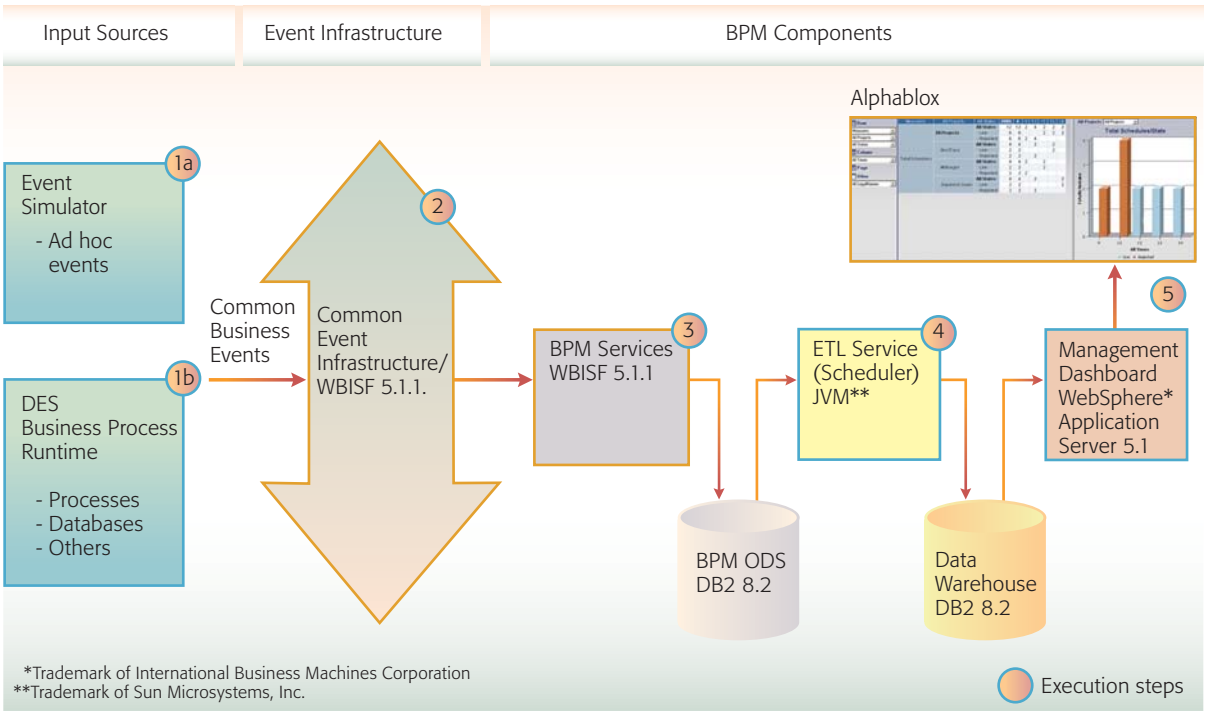

Figure 3
BPM runtime system deployment

robust by implementing the solution on many IBM Research projects, such as Distributed Enterprise Services (DES) and On Demand Distributed Computing Services (ODCS). MDD BPM is still in an early stage of development, but is expected to continue gaining wider acceptance. The work we are presenting is part of the larger Model Driven Business Transformation (MDBT) toolkit effort within IBM Research. The MDBT toolkit with instruction documents is available to download.

Related work

Due to its high-level abstraction and code reuse feature, the MDD methodology has been widely applied in related areas such as software reuse,[18,19] reverse engineering,[20] and user interface design.[21] The benefits of adopting MDD include reduced software development time, enhanced code quality, and improved code maintenance.[10,22]

There are also numerous related works about business processes. Widely considered as an extension of a workflow management system, business process management enables the management and analysis of operational business processes.[23] Most recent work has focused on modeling business processes, consistency checking for model integration, and composing Web services and business processes by using the model-driven approach.[24–26] Recently, model transformation has received much attention because it can bridge between source models and target models.[22,27] Though there are many standards defining individual models, there is no one model transformation standard. Domain-specific languages[28–30] and UML profiles have been defined and used to express the transformation logic or mapping rules.

Our work focuses on a generic model-driven framework that aims at code customization, code reuse, merging multiple models, and constraint validation. Due to the well-defined BPM observation metamodel and the many strict constraints imposed by it, we chose to deal with the model transformation and mapping the code after the code-generation phase for both the source model and target model. We used the well-developed Eclipse Modeling Framework (EMF)[31] to generate model manipulation code for instantiating in-memory model instances, which have strong built-in validations.

To better understand our technology framework, we used the IBM DES project as an example and a generalized scenario.

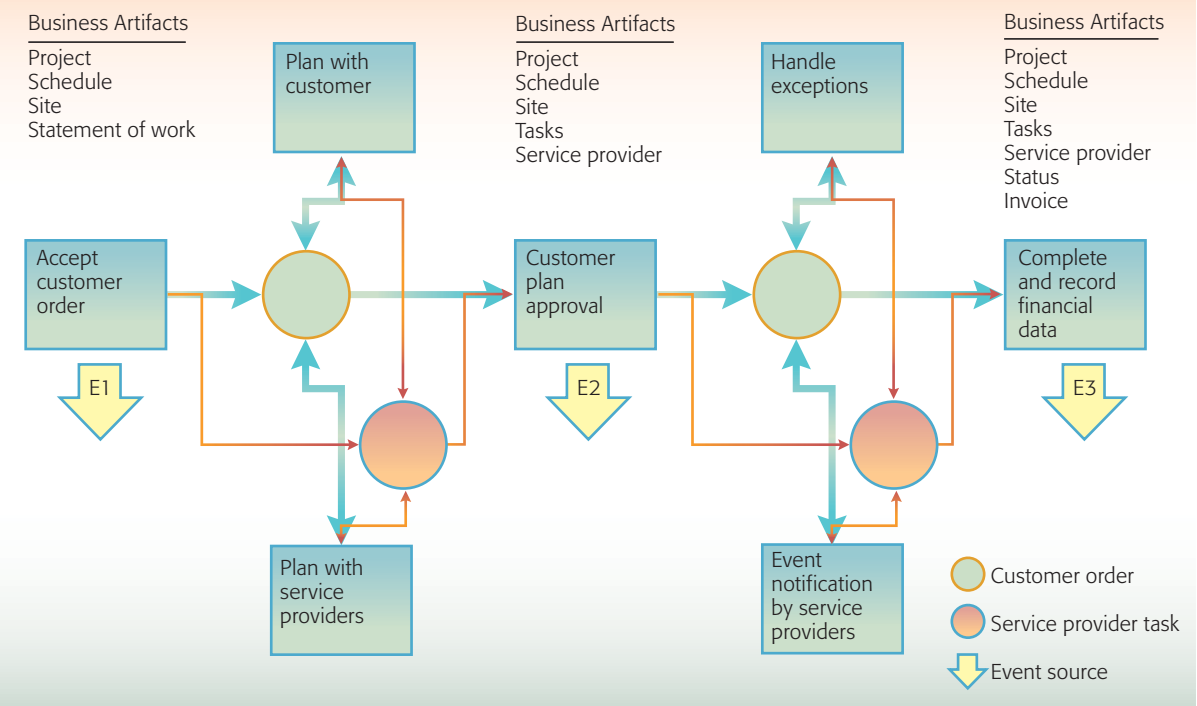

Figure 4
DES service-delivery operations model

SAMPLE SCENARIO

The IBM DES pilot project was the first to demonstrate the BPM solution using UML models and MDD techniques. We describe DES and then a sample scenario known as *service delivery*.

Large enterprises with a number of relatively homogeneous but independently operating sites at which the central value of the enterprise is created are referred to as *distributed enterprises*. Examples are national retail chains or banks with many branch offices. Each property operates in many ways as an independent business but with varying degrees of central ownership, shared resources, and business function. Such enterprises have a project management office (PMO) that centrally serves the needs of the distributed customers, promotes the efficient use of resources, and manages costs by using economies of scale. This scenario is called the *service delivery model*, and a generalized business process for such a model is shown in ***Figure 4***. The business operations were defined as an artifact-based model,[32,33] a business modeling technique particularly suited to direct business goal mapping and business integration. The operational model of the business consists of the steps required to achieve operational goals and the flow of business artifacts through them.

The goal of delivering a complete service to a customer site on an agreed-upon schedule is captured in the DES operational delivery model. Actually, it involves two interacting sets of goals: satisfying customer needs and performing specific services to that end. This leads to three primary artifacts:

1. *Schedule*—An attachment to a statement of work for services and equipment delivered to a particular site. The schedule is the primary artifact for customer delivery interactions. Services defined in the schedule include a delivery plan organized by task. The task information maintained in the schedule represents units of work required to complete the schedule, including service provision and equipment delivery.
2. *Supplier task*—A unit of work performed by a service provider.
3. *Project*—A set of schedules.

Process monitoring, measurement, and metric information can be obtained by monitoring the

process events generated at probe points within tasks and repositories. Figure 4 shows the event source probe points E1, E2, and E3. These events must contain sufficient information for correlation and metric calculation. The BPM technique can be used to provide sophisticated real-time notification of complex patterns of events. An entire suite of composite metrics can be updated at the end of each business event.

GUIDELINES FOR IMPLEMENTING A BPM SOLUTION

Based on our experiences with DES and several other projects, we have developed a set of guidelines that can ease the task of designing and developing a complex BPM solution (*Table 1*). Although the guidelines are expressed as six steps, we recognize that it might take several iterations of design and development before an optimal level of solution maturity is achieved and business monitoring requirements are met. It is advisable to start with a project of small scope with few metrics, but important enough for stakeholders to measure the success of BPM technology.

Step 1: To gather requirements, the end user of the solution and the sources of the input data for the BPM runtime system are determined, and the reports that the end user would like to see on the dashboard are identified. It is also necessary to identify the metrics (business goals) that the end user wants to monitor, the business conditions that must be detected, and the actions that may need to be taken. Some metrics may be related to one another; others may need sophisticated calculation. Gather all such information at this stage. The input data sources might be well-defined business processes that emit (generate) business events, existing data warehouses, or operational databases. It is important that an agreement is reached with the stakeholders regarding the business goals and how these goals are defined and calculated.

In the DES scenario, the input data sources are business events. The end users are of two types, operational and executive. For operational metrics, we selected the following key performance indicators (KPIs):

- Total number of schedules or tasks by status (planned, live, completed, or canceled)
- Total outstanding schedules or tasks at the project level

Table 1 BPM solution guidelines

BPM requirements and goals	Step 1	Requirement gathering (business process, data, goals, reports)
	Step 2	Analyze and transform requirements (event data, metrics, KPIs, context, rules, views)
BPM Platform-Independent Model	Step 3	Map the requirement to models and their constructs (e.g., OMs and monitoring contexts)
	Step 4	Provide additional model-related information (metric calculation, outbound event, data warehouse needs)
	Step 5	Model transformation into intermediate models (data warehouse model, runtime model, view model)
BPM Platform-Specific Model	Step 6	Platform-Specific Models and deployment (OM-runtime code, data warehouse schema, ETL, event emitters)

- Time span between when schedules and tasks are planned and their actual completion time
- Total time between plan and completion at the project level

Executive-level metrics might be total service-level agreement (SLA) violations and the total number of pending schedules. Metric results can be displayed on the dashboard in real time and one can probe more deeply for finer detail and see broader aggregate views.

Step 2: The requirements are analyzed to identify the elements needed for the BPM model. These might include events, KPIs, metrics, management contexts, business conditions, and reports. One needs to identify appropriate metrics and their relationships with other metrics and the management context.

In the case of the sample scenario, the events identified were `ScheduleEvent` and `TaskOrderEvent`. The management contexts identified were `bySchedule`, `byTaskOrder`, and `byProject`. Similarly, metrics and business conditions need to

be defined without ambiguity because the BPM solution is modeled based on these elements. Later, we show the BPM elements in a requirement model.

Step 3: BPM requirements are modeled using RSA with extended UML2 profiles for BPM elements. For clarity, it is a good idea to create an individual model for each management context identified in Step 2. One might need to go back to Steps 1 and 2 while modeling the solution. The models created in this step are the OMs. The details of how models are created are given in subsequent sections.

Step 4: To augment the OM, the expressions that are needed to calculate metrics or to add business conditions or outbound events in response to changing business situations are provided. Also, it is determined whether there is a need for data warehousing. If so, the appropriate data warehouse model needs to be created, based on the OM created in Step 3.

Step 5: The transformation of the models is performed by selecting the transformation menu in RSA. The transformation results in the generation of intermediate models in the output folder. Usually one does not need to perform any updates at this stage, but if there are certain complex needs that could not be addressed by means of our model framework, this would be the time to reflect those changes in the intermediate models. This step may also be of use if non-modeled BPM solutions are transformed to our intermediate models and there is a need for updating.

Step 6: From the intermediate models, the Platform-Specific Model, code, and deployment scripts are generated. The person managing the deployment then deploys the various components in the appropriate environment (Figure 3).

BPM MODELS

In the previous sections, we introduced our approach at a very high level and provided guidelines to define and use BPM modeling. In this section, we discuss core BPM models, such as the OM and the data warehouse, in detail with the help of our DES sample scenario.

Observation model

This section is divided into four parts: general terms the BPM elements use to create OMs, definition of the requirement model using the DES sample scenario, a brief discussion on mapping the requirement model to the OM, and a discussion on optimizing the OM for execution.

Elements and OM detail

The main modeling elements are shown in Figure 1. A core model element is `ManagementContext`. Each `ManagementContext` element represents a business artifact that needs to be monitored and controlled or managed. A business artifact may be monitored at either the instance level (e.g., the processing time of individual customer orders) or at the aggregate level (e.g., the average processing time of all customer orders). If only instance-level monitoring is required, one `ManagementContext` element may suffice to define all observables (metrics) for the same kind of monitored entity. If aggregate-level monitoring is desired, multiple `ManagementContext` elements may need to be defined, representing the artifact instances and different granularities of the artifact aggregate, respectively.

A `ManagementContext` element encapsulates the state and behavior of the managed artifact. The state in a `ManagementContext` element consists of a collection of metrics. Different instances of the same `ManagementContext` element are uniquely identified by a key metric. Metrics are typed and may be computed via maps. There are two special kinds of metrics: *situations* and *timers*. Situations are Boolean-type metrics that define business conditions warranting actions or attention. Timers are a special kind of metric that behaves like a stopwatch. The value of a timer is updated when the timer is started, stopped, or reset.

A `ManagementContext` element subscribes to `BusinessEvents` that report state changes of the managed entity. Such a subscription is specified in an `EventContextLink`, which defines the filtering and correlation constraints for `BusinessEvents` of a particular type that are received in a `ManagementContext` element.

The state of a `ManagementContext` element may be mapped to elements in a data warehouse for online analytical processing. A `Dimension` defines a dimension table in the warehouse, which could be preexisting or created from scratch. A `MeasureDimensionLink` maps a numeric metric (measure) to a column in the dimension table. An `AttributeDimensionLink` maps a categorical metric

to a column in the dimension table for the purpose of populating the column with the metric value.

DecisionMaps and BusinessActions specify the control on managed entities. They are part of a ManagementContext and are triggered by Situations. A DecisionMap selects among alternative BusinessActions. A BusinessAction is a logical container for actions to be undertaken to resolve the situation at hand. Payoff Functions can be associated with a BusinessAction to calculate the cost of executing the action in terms of money or duration.

ManagementContext elements can form a parent-child ContextRelationship. A parent context may represent a "superordinate" entity or an aggregate. A parentKey map dictates how the key metric of the parent context can be computed from the state of the child context.

Requirement model using sample scenario

From the BPM requirement point of view, the most important question one can ask is, "*What do we want to monitor and what notifications do we want to receive?*" It could be as simple as monitoring items such as revenue and cost or tracking a forecasted revenue against an actual revenue and having an e-mail sent when a monitored metric violates a threshold. For our sample scenario, the four KPI artifacts we identified were listed earlier under Step 1.

Other business artifacts, such as sites, task orders, schedules, and time, characterize these monitoring metrics. The expressions used to calculate the monitored metrics must also be known. (Each characteristic could lead to multiple combinations with monitored artifacts for measurement purposes, but only a few are deemed important from the user perspective.) We identified three management contexts for our sample scenario: bySchedule, byTaskOrder, and byProject. The following input data, from an underlying business process system, flows into the BPM system in the form of the business events: ScheduleEvent and TaskEvent.

Figure 5, Part 1 represents the two management contexts bySchedule (child) and byProject (parent–aggregation). This figure follows the guidelines in the IBM business-observation-metamodel specification and extensions explained in the previous section. Shown are the monitored metrics, events, maps, and other elements that were identified during Steps 1 and 2.

Conditions can be monitored and defined accordingly. For instance, in the bySchedule management context, the SLASituation element was identified to represent the business situation if the condition TimeDiff metric crosses a threshold value. If the condition is set to true, then an outbound event called SLAViolationEvent is generated.

The other management context could be represented similarly to Figure 5, Part 1. Such graphic representation of the models plays an important role in creating OMs by using RSA and UML2 profiles.

Sample scenario: How to create a UML OM

We now discuss in brief how requirements are actually modeled. We assume the BPM profiles have been imported into the model workspace and that a general user has a basic understanding of using UML and profiles,[12] as it is beyond the scope of this paper to review standard language detail. For simplicity, we model only one management context: bySchedule.

Figure 5, Part 2 shows the sample scenario representing the bySchedule OM by using UML2 profiles. We refer to Figure 5, Part 1 while we create the UML Model in the following five steps:

Step 1: Define ManagementContext class. Using the RSA model editor,[8] we start by defining a class that represents the management context bySchedule. Next, we apply the stereotype <<ManagementContext>>.

Step 2: Define BusinessEvent class. Define a class called ScheduleEvent and stereotype it as <<BusinessEvent>>. The BPM runtime will receive these events as input. There is a simple association link stereotyped as <<EventContextLink>> between the ScheduleEvent class and the bySchedule management context. The attribute of this stereotype determines the system logic for the incoming event during the runtime. The attributes are noCorrelationMatches = 0(createNewContext), oneCorrelationMatch = 2(deliverToAny), and multipleCorrelationMatches = 4(raiseException). It also has a business rule constraint of type event

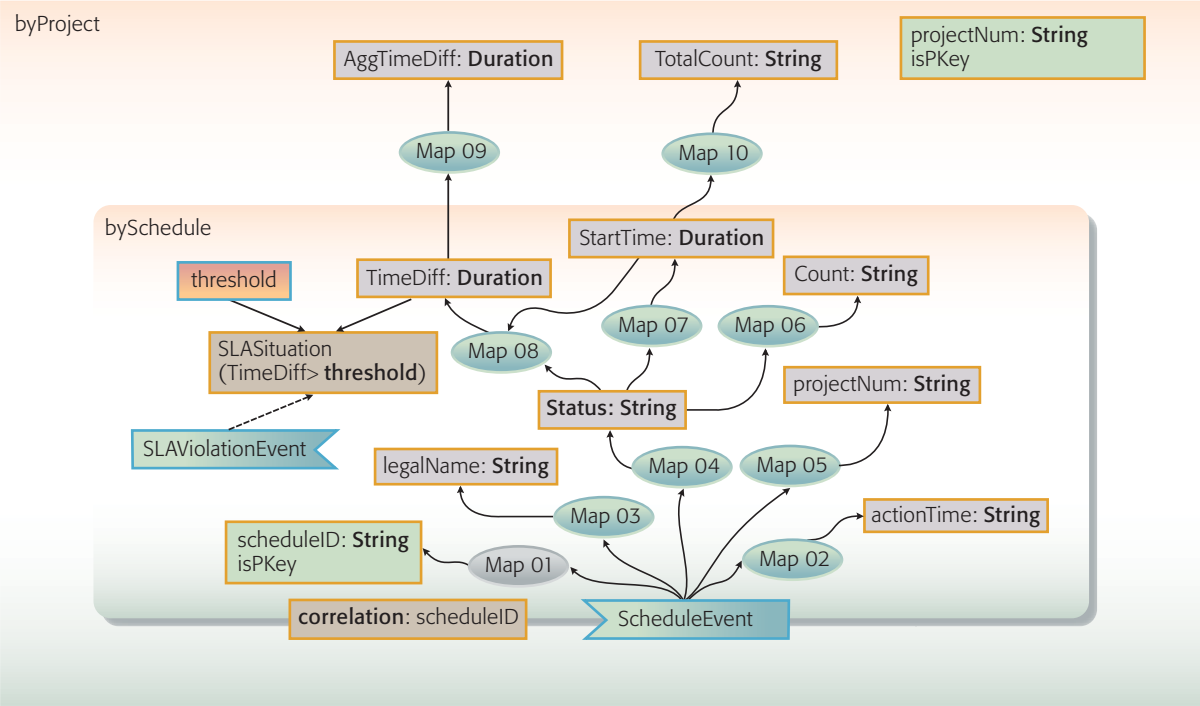

Figure 5
DES sample scenario; Part 1 of 2: observation model view

correlation that determines the correlation predicates for creating a new management context instance in the runtime.

Step 3: Define Metric class and mappings. Define all classes for the monitored metrics and stereotype them as <<Metrics>>. Figure 5, Part 1 is an excellent reference for the list of monitored metrics that need to be modeled. Each metric class has an attribute called "value", and its type determines the data type of the attribute. At this point a composition association is defined between the metrics and the management context, as shown in Figure 5, Part 2. The stereotype metric has properties that need their values to be populated, such as `keepHistory`, `multiplicity`, `partOfKey` and `readOnly`. If there is a need to maintain the history of any monitored metric, then the value of `keepHistory` must be set to 1. In Figure 5, Part 1, `scheduleID` represents the part of the correlation key (`ispkey=true`) for the current management context; hence, the `partOfkey` property is set to `true`. To calculate the value of the metric, an operation is added and stereotyped as <<Map>>. On this map, add the UML element `ReturnResult`. From the properties of `ReturnResult`, select the Create New expression and enter the expression in the general body of the element. The evaluation expression could look like `current time - bySchedule.startTime.value`. One also needs to define a trigger condition for this map to be evaluated. One such condition could be the arrival of the business event with a status metric value of `actual`. (One needs to refer to the BPM profiles document to understand the expressions supported in the current release.)

Step 4: Define Situation class. Create a class called `SLASituation` and stereotype it as <<Situation>>. Next, define the gating condition for each situation. Add an operation, for example, `condition`, and stereotype it as <<Map>>. Specify the expression and evaluation trigger for this operation. The expression forms the `ReturnResult` part of the operation. Once `condition` is evaluated to `true`, an outbound event called `SLAViolationEvent` is generated. Within the `SLASituation` element, add another UML constraint called post-condition and name it `SLAViolationEvent`. One needs to specify the metrics that will form the data part of the outgoing event in this constraint.

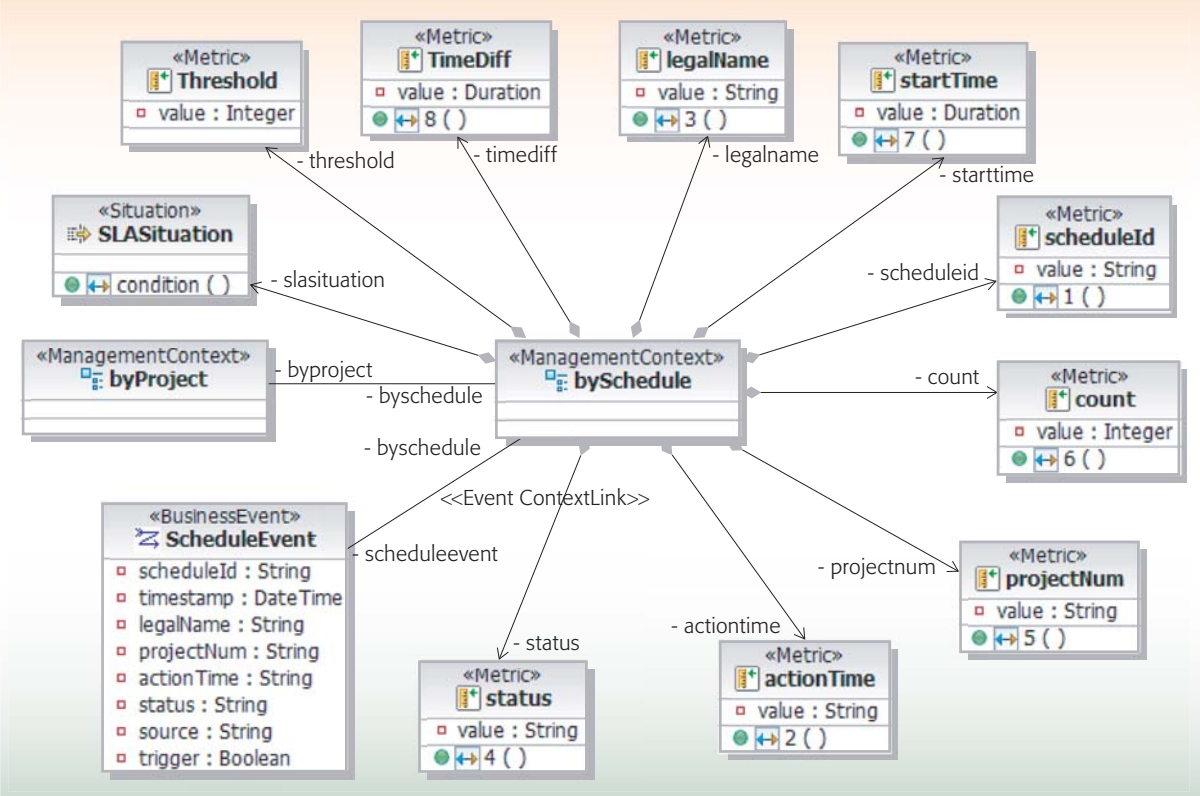

Figure 5
DES sample scenario; Part 2 of 2: bySchedule observation model using UML2 profiles

Step 5: Define appropriate links with other elements. Define all the other `ManagementContext` elements similarly. Then, each `ManagementContext` element should be linked to its parent contexts through a directed association stereotyped to `ContextRelationship`, as shown in Figure 5, Part 2. For each relationship, the following stereotype properties also need to be specified: `parentContextAutoCreated`, `parentContextMandatory`, `parentContextTerminationCascades`, and `Primarykey`.

Model-driven adaptive data purging

The OM describes the processing path of inbound events and the resulting actions. In particular, it includes filtering conditions for business events based on which events are considered relevant with respect to possible management contexts. Without the knowledge derived from the OM, general filters need to be placed in the network at the emitters or in the event bus. The monitor subscribes to events that pass these filters. Clearly, several problems can result from this approach. They include lack of scalability with respect to event sources and monitors, contention at the event databases during event storage and query, and, in general, inefficient use of network and computational resources.

To address these shortcomings, we developed a technology for model-driven adaptive data purging. The advantage of this approach is that it automates the placement of filters throughout the network and restricts event flow to the relevant events. Most importantly, we are now able to automatically place the right filters on the right components. These filters are derived from the OM before its activation and are directly relevant to the active monitors. The following steps are necessary for adaptive data purging:

- Extract filtering conditions from the OM.
- Decide on the semantics and the placement of subscriptions in the system. The conditions extracted in the previous step are analyzed against

knowledge about the topology, available resources, and the processing capabilities of components.
- Communicate the subscription plan to components and ensure that they are able to process the assigned subscriptions.
- Ensure subscription plan validity.
- Activate subscriptions.

Without model-driven adaptive data purging, the knowledge from events flows from emitters to the consumer, which is the OM runtime system. Our technique allows for knowledge of the OM used by the monitor to flow back upstream in the event flow to facilitate the placement of tighter filters on the right components of the system.

Data warehouse metamodel

Data warehouses are crucial as a store for historical artifacts and to support analytics. Their design is a highly disciplined skill; data warehouse[34,35] creation takes considerable time and effort and is mostly a manual process. Nonetheless, many times the data represented in a data warehouse is not connected to or directly representative of business models, and it can be difficult for stakeholders to analyze the data. Such designs are also not adaptive; with changing business models, redesigns are required. With the advent of BPM came a need for a data warehouse that can adapt in real time.

To automate the process of creating a simple and adaptive data warehouse and to also preserve linkages with business models, a data warehouse metamodel is proposed. We briefly describe both the data warehouse and schema generation in the next subsections. We also define other artifacts generated for analytics purposes, such as IBM DB2 Cube Views[36] models for OLAP,[34] which are used for default dashboard generation.

The data warehouse metamodel (DWMM) provides for capturing specific information about the structure of the data warehouse and the semantics of the business models that it represents. Thus, an instance of the metamodel is a DWMM that represents information at two levels of abstraction: at the logical level, the model keeps information about all the measures and dimensions in the warehouse and their interrelationships; at the physical level, the model supplies details about the specific physical representation of the measures and dimensions in the database. Note that the logical part of the metadata model contains enough information for the automatic generation of the physical part, which is populated at schema-generation time.

Using XML Schema, the high-level structure of the DWMM is as follows. The root element `bpmschema` contains a sequence of four subelements that represent types of information. At the logical level, dimensions are represented by `DimensionDefinition`, information about the measures is stored in `MeasureDim` elements, and the relationships between measures and dimensions are specified by `MeasureToDimension` elements. At the physical level, the elements of type `MetaFactTableGroup` store details about the physical representation of measures in tables. The information related to the business models is also stored within the subelements, providing the bridge between business model elements and data artifacts in the data warehouse.

UML model

Figure 6 shows another BPM model that represents the DWMM. To begin building this model, select the metrics (defined earlier in Figure 5, Part 2) that are of interest to the storage purposes in this view. To build this model, one needs to understand *dimensions* and *facts*. A *dimension* is a group of metrics related by some hierarchy; for example, for the time dimension, day, month, and year are related as parent–child. A *fact* is a metric that is measurable. For analysis, a fact metric is meaningful with a context; for example, revenue for a given product by month. In this case, revenue is a fact metric, and product and month are dimension attributes. The steps in the design of the BPM data warehouse model proceed with this concept in mind.

Step 1: Create a Dimension class and metrics as its attributes. This step identifies the groups of metrics that are related, creates a class, and stereotypes it as <<Dimension>>; for example, `ScheduleDim` as shown in the figure. The stereotype has attributes such as `isExisting = false`, which means a new dimension (dynamic), and `isPopulatedAtRunTime = true`, which means that the source of data in this dimension table will be the BPM runtime operational data store. Then the attributes for the dimensions are defined, for example, `scheduleID`, `projectNum`, and `legalName`, and stereotyped as <<DimensionLevel>>. This stereotype has an

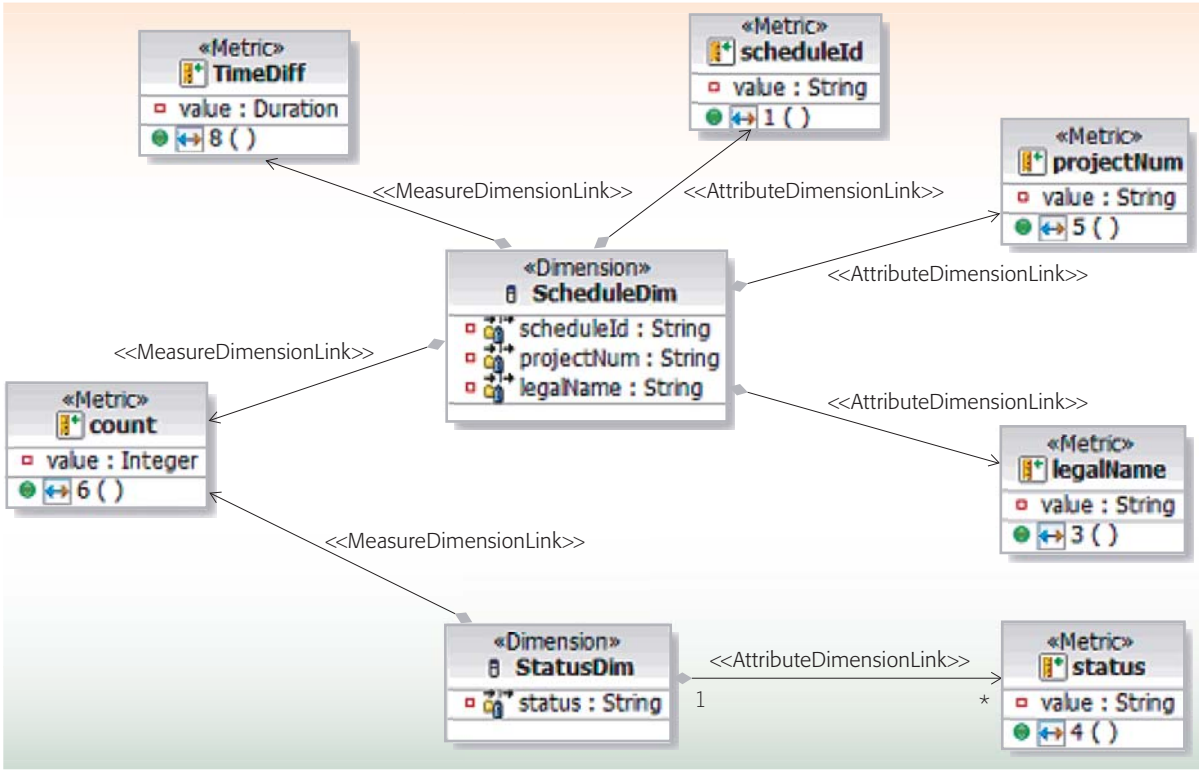

Figure 6
BPM performance-warehouse model using UML2 profile

attribute called *level*. The level is set starting at 0 for the attribute that forms the leaf node in the hierarchy. For example, the level for `projectNum` is 1, and the level for `legalName` is 2. Each dimension needs to have a primary key. For `scheduleDim`, `scheduleID` is the primary key; hence, the <<PrimaryKey>> stereotype is also applied to this attribute.

Step 2: Create dimension-attribute link with metrics. This step creates a directed link from dimension (`ScheduleDim`) to the metric class called `scheduleID`. This link is stereotyped as <<DimensionAttributeLink>>. This stereotype has an attribute called `attributeName` that represents one of the attributes of the `ScheduleDim` dimension, such as `scheduleID`. There is another attribute for DimensionAttributeLink called `correlationID`, which correlates sets of metrics that represent the same attributes in the dimension. This link helps in populating the dimension table at runtime. The `DimensionAttributeLink` is not required if a dimension already exists and does not need to be populated at runtime. Add the directed link from `ScheduleDim` to other attributes and sterotype it as <<DimensionAttributeLink>>. Create other dimensions as needed, for example, `StatusDim`.

Step 3: Create measure dimension link with metrics. This step identifies the measure (fact) or monitored metric. In this example, the `count` and `TimeDiff` metrics are identified as measure metrics. A directed link is created from the measure metric to the dimension class `ScheduleDim` and stereotyped as <<MeasureDimensionLink>>. This stereotype has the attributes `bywhichDimensionAttribute = scheduleID`, an attribute of a dimension, and `referenceMetric = scheduleID` as the metric. This defines the measure-to-dimension link that helps in determining the relationships between the fact table and the dimension table as part of star schema generation. The measure metrics are linked to other dimensions as determined by the analytics.

As the model is created, the business metrics are used to link the data-warehouse-related schema elements. Hence, the performance DWMM captures

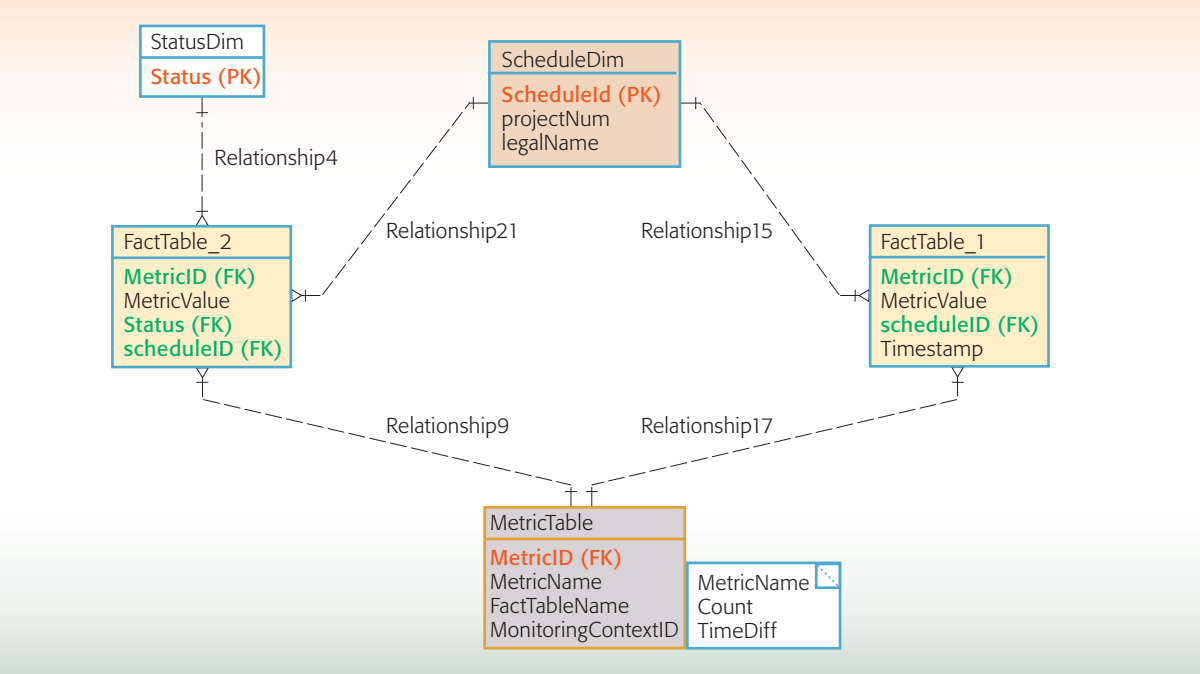

Figure 7
BPM performance-warehouse physical model

both business metrics and data-warehouse-element semantics, which is helpful in managing the data warehouse with changing business metrics.

Star schema generation

The schema-generation algorithm for the star schema form clusters of measures by the set of dimension attributes to which they are linked and generates a fact table for each resulting cluster. For increased flexibility, we decided to use a vertical schema representation for the measures, which accommodates the subsequent addition, renaming, and removal of measures without needing to alter the table definition. Thus, all measures in cluster C_i are stored in a single fact table with the following structure: FactTable$_i$(MeasureID, MeasureValue, RD_1, RD_2, . . ., RD_k), where MeasureID stores a numerical identifier for the measure, MeasureValue stores the measure value (a double-precision number), and RD_1, RD_2, . . ., RD_k represent references to the dimension tables D_1, D_2, . . ., D_k. It is important to note that the RD_k terms do not have to reference the primary key of their respective dimension table; they can reference an arbitrary column (or group of columns) in the dimension table. *Figure 7* shows the data warehouse schema generated during the transformation of the BPM data warehouse model shown in Figure 6.

ETL process

ETL is the process of analyzing the incoming data from the ODS to make it suitable for storage in the data warehouses. In the case of BPM, the ODS is the BPM runtime data store where metrics information is stored as it gets created or updated. The ETL process is the mix of Java** programs and Structured Query Language (SQL) scripts. The Java program is static in nature and takes the model-generated SQL scripts as input and executes them at appropriate intervals to populate the dimension and fact tables. These SQL scripts are autogenerated by the Java program during the model transformation process by taking the data warehouse model instance, which is created during the model transformation phase, as input.

Cube model

For the historical and multidimensional analysis, OLAP solutions are the best available tools at the enterprise level. IBM published a cube view metamodel[36] to represent multidimensional information in an intermediate metadata format. IBM also made available various adapters and techniques to export

Table 2 Sample DES events as sent to the BPM runtime server

scheduled	timestamp	legalname	project_number	actionTime	status	source	trigger
SC111	2005–12–10–21.12.14.658001	MyFoodInc	B111	2005–12–10–21.12.14.658001	Plan	business process	0
SC112	2005–12–10–21.12.14.658002	MyFoodInc	B111	2005–12–10–21.12.14.658002	Plan	business process	0
SC111	2005–12–14–21.12.14.658003	MyFoodInc	B111	2005–12–14–21.12.14.658003	Accepted	business process	0
SC112	2005–12–12–21.12.14.658004	MyFoodInc	B111	2005–12–12–21.12.14.658004	Rejected	business process	0
SC111	2005–12–15–21.12.14.658005	MyFoodInc	B111	2005–12–15–21.12.14.658005	Live	business process	0
SC111	2005–12–21–21.12.14.658006	MyFoodInc	B111	2005–12–21–21.12.14.658006	Complete	business process	0

this metadata to IBM Alphablox and to other OLAP systems, such as those by Cognos and Hyperion Solutions Corporation. The BPM solution provides an automated routine that transforms the model artifacts in the data warehouse model to cube view metadata. This enables the BPM analytics to be installed on popular OLAP engines, such as Hyperion System 9 BI+**, Microsoft SQL Server, and Alphablox, and allows users to perform more in-depth analysis of the data to detect trends and patterns.

Model integration and sample scenario execution

Figure 3 shows a deployment setup used for the DES sample scenario. The DES artifacts-based business process acts as the source for incoming business events. The BPM runtime components[37] generated from the sample scenario models are deployed in the BPM services block, and the BPM ODS schema is deployed as BPM ODS. The data warehouse schema and ETL service are deployed in their respective blocks. The data-warehouse Alphablox OLAP application is installed in the management dashboard block.

Once the BPM components are deployed, the event simulator, which simulated the process events to start the execution of the BPM system, emits the events (**Step 1a**; the steps in this section refer to Figure 3). *Table 2* illustrates a few sample events that are sent to the BPM runtime server. These events are related to the life cycle of schedules or orders in the process manager. In production, these events will be replaced with events coming from actual processes (**Step 1b**).

The event data is converted to the CBE (common base event, sometimes called common business event) format and then published to the event infrastructure, which comprises the CEI runtime setup (**Step 2**). The events are then forwarded by the CEI to the event subscribers. These are the BPM services, which are also the consumers of the events (**Step 3**). BPM services process these events appropriately and save runtime data in the ODS. The ETL service is scheduled to run every 5 minutes (**Step 4**) to extract, transform, and load the data into the data warehouse. The BPM dashboard then pulls the data from the data warehouse per reporting requirements or queries and displays it on the dashboard (**Step 5**). In case of DES, we used DB2 Alphablox and its OLAP engine to display the data in a multidimensional format to enable the analytics for the business user. As events are published, processed, and stored in the data warehouse, the Alphablox display reflects the changes.

LESSONS LEARNED

The MDD approach provides a bridge between business and IT and provides a rapid-solution development platform. It also provides flexibility in adopting changes as business processes evolve.

In the initial phase of the MDD platform, we had BPM OM for capturing monitoring elements and automatic code generation for OM runtime only, not for data warehouse or action management, due to a limitation in the OM specification. For the DES solution, we had to manually create the data warehouse and hence, lost the business metrics mapping with the data elements in the data warehouse. It also took a great deal of time and much cost to manually create the data warehouse. We had to rely on the expertise of a business analyst and a data architect to bridge the gap and create the custom dashboard to display the information. Even

the manual effort of developing the warehouse schema and ETL took a number of iterations because translating the model information into the physical data warehouse schema was a complex task. Hence the need for the extension of the OM was determined; Figure 1 represents the extension.

With regard to the UML2 profiles, the BPM editor itself had to undergo a few iterations of change as the functionality in the initial version was not sufficient to support the DES solution requirements. We learned that MDD for BPM is an evolutionary technology and that application requirements often drive the expansion of BPM tools. This also became evident when we determined that there was a need to access external data sources within the OM and, as a result, the BPM tools and transformation logic had to evolve.

A right model cannot be created the first time. It takes a few iterations before a model is deemed suitable for the solution. This was a very important lesson as we had several components, such as the data warehouse and action components, that required extensive resources to develop. Hence, we identified the need to autogenerate the data warehouse and action components. This automation was completed soon after. (The details are in the earlier section on the BPM data warehouse metamodel. A discussion of the autogeneration of the action components is beyond the scope of this paper.)

The function of the BPM runtime components to execute the OM and data warehouse components depends upon how well the BPM problem is modeled. Hence, it is very important that the solution is modeled correctly. In the initial DES models, the business analyst had left a few artifacts in the model that were unused, and the ETL component failed to load the data into the data warehouse as it was not getting propagated for such orphaned artifacts.

Another important issue we encountered in developing the DES solution was model validation. Before platform-independent models are to be transformed into platform-specific models, they need to be verified and validated. We can use the model validation component to help users locate potential problems in their models. However, there are still areas where the models defined by users seem correct, but cannot be processed by the transformation engine due to the fact that openness is deliberately engineered into the platform-independent models for the sake of modeling flexibility at the business level. Increasing the precision of the platform-independent model tends to limit flexibility for business users. Hence, there is a trade-off between flexibility and precision at the levels close to business semantics. The decision about the degree of flexibility and precision will imply the degree of difficulty of validating models before transforming them into platform-specific models.

BPM models can become complex, as can runtime components. This could lead to a solution that may not perform optimally. We continue to research new ways of improving the performance of the BPM runtime system, and one such solution, adaptive data purging, is currently under development. Adaptive data purging is one step in increasing the scalability of a BPM solution. The current implementation uses knowledge derived from the OM at design time. The processing of an OM, however, has a dynamic aspect, which is the creation of monitoring contexts. This limits the scope of our event-filtering techniques. In extending our solution, it is important to consider the dynamic knowledge extraction from the monitor runtime.

The UML2 profile was not the first approach to representing the OM in a modeling tool. We started with an XML metamodel to represent the OM for the BPM solution. Because XML is human-readable, it was suitable for the smaller BPM solution. Figure 2 shows the model flow for both XML and UML2 approaches.

As this paper was being prepared, BPM models were used by other projects such as ODCS and Telesales World Wide Dashboard. We learned from user experience that BPM models are quite engaging and that there is a definite learning curve. Because the solution is model-driven, the success of the application is as good as the model created by the business users; hence, one should be prepared to invest time to understand how to design a BPM solution using UML2 BPM profiles.

CONCLUSION

In general, MDD provides flexibility in adapting to changes as business processes evolve, and its use results in considerable time and cost savings. BPM solutions are usually complex in nature and take

considerable time and money to develop. With the autogeneration of runtime components, one can experiment with creating models and see the results very quickly by deploying the generated components. In our DES prototype, it was very helpful to use this feature, as the monitoring requirements were in constant flux before the system started to stabilize. In general, monitoring requirements in the real world are never stable. The benefit of the BPM MDD approach was realized with the adoption of this technology on the projects we have conducted.

As adoption of this technology spreads within and outside IBM, the modeling requirements become more complex, and the BPM models and runtime components evolve. Presently, if a model is changed, then the runtime components need to be completely redeployed. We are working on the model to develop components that can take incremental changes during runtime and maintain the existing runtime environment. We are also working to identify new models, such as report models, that could interact with our BPM models so that customized reports can be generated automatically. We remain enthusiastic that MDD techniques provide enough benefit that they will be widely adopted in the BPM area.

ACKNOWLEDGMENTS

We thank Santhosh Kumaran, Prabir Nandi, Terry Heath, Kalyani Deshpande, and Kamal Bhattacharya for their work on artifacts-based operational modeling work for DES. We also thank our colleagues who are involved in the development of the MDBT toolkit and our colleagues at the IBM Software Development Laboratory, Taiwan, especially Jimmy Tan and Sam Wang, for their work in developing the DES BPM dashboard.

CITED REFERENCES

1. IBM Software Group, *Establishing a Business Performance Management Ecosystem*, White Paper, IBM Corporation, Somers, NY, ftp://ftp.software.ibm.com/software/integration/pdf/bpm_whitepaper_0301.pdf.
2. C. Ballard, C. White, S. McDonald, J. Myllymaki, S. McDowell, O. Goerlich, and A. Neroda, *Business Performance Management . . . Meets Business Intelligence*, IBM Redbooks, IBM Corporation (August 8, 2005), http://www.redbooks.ibm.com/redbooks/pdfs/sg246340.pdf.
3. H. Dresner, "Business Activity Monitoring: BAM Architecture," *Gartner Symposium ITXPO*, Cannes, France (2003), http://www.pikos.net/documents/german/Gartner.pdf.
4. S. H. Haeckel, "Leading On Demand Businesses—Executives as Architects," *IBM Systems Journal* **42**, No. 3, 405–413 (August 2003).
5. S. H. Haeckel, *Adaptive Enterprise: Creating and Leading Sense-and-Respond Organizations*, Harvard Business School Press, Cambridge, MA (July 1999).
6. P. Chowdhary, L. An, J.-J. Jeng, and S.-K. Chen, "Enterprise Integration and Monitoring Solution Using Active Shared Space," *Proceeding of the IEEE International Conference on e-Business Engineering,* Beijing, China (2005), pp. 665–672.
7. S. Kapoor, K. Bhattacharya, S. Buckley, P. Chowdhary, M. Ettl, K. Katircioglu, E. Mauch, and L. Phillips, "A Technical Framework for Sense-and-Respond Business Management," *IBM Systems Journal* **44**, No. 1, 5–24 (2005).
8. IBM Rational Software Development Platform, IBM Corporation, http://www-306.ibm.com/software/info/developer/busvalue.jsp.
9. Borland® Together™ technologies, Borland Software Corporation, http://www.borland.com/us/products/together/index.html#architect.
10. A. Kleppe, J. Warmer, and W. Bast, *MDA Explained, The Model Driven Architecture: Practice and Promise*, Addison-Wesley, Boston, MA (2003).
11. S. Kumaran, "Model-Driven Enterprise," *Proceedings of the Global Enterprise Application Integration Summit,* Banf, Canada (2004), pp. 166–180.
12. UML2 Profiles, The Eclipse Foundation, http://www.uml2.org/.
13. S.-K. Chen, H. Lei, M. Wahler, H. Chang, K. Bhaskaran, and J. Frank, "A Model Driven XML Transformation Framework for Business Performance Management," *Proceedings of the IEEE International Conference on e-Business Engineering,* Beijing, China (2005), pp. 71–78.
14. *XML Metadata Interchange (XMI) Specification, Version 2.0*, Object Management Group Inc., http://www.omg.org/docs/formal/03-05-02.pdf.
15. M.-T. Schmidt, B. Hutchison, P. Lambros, and R. Phippen, "The Enterprise Service Bus: Making Service-Oriented Architecture Real," *IBM Systems Journal* **44**, No. 4, 781–797 (2005).
16. Common Event Infrastructure, IBM Corporation, http://www-306.ibm.com/software/tivoli/features/cei/.
17. DB2 Alphablox, IBM Corporation, http://www-128.ibm.com/developerworks/db2/roadmaps/alphablox-roadmap.html.
18. W. B. Frakes and K. Kang, "Software Reuse Research: Status and Future," IEEE Transactions on Software Engineering **31**, No. 7, pp. 529–536 (2005).
19. J. Greenfield, K. Short, S. Cook, and S. Kent, "Software Factories Assembling Applications with Patterns, Models, Frameworks and Tools," *18th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems,*

Languages, and Applications, Anaheim, CA (2003), pp. 16–27.

20. S. Rugaber and K. Stirewalt, "Model-Driven Reverse Engineering," *IEEE Software* **21**, No. 4, pp. 45–53 (2004).
21. N. Sukaviriya, S. Kumaran, P. Nandi, and T. Heath, "Integrate Model-Driven UI with Business Transformations: Shifting Focus of Model-Driven UI," *Proceedings of the Workshop on Model Driven Design of Advanced User Interfaces*, Montego Bay, Jamaica (2005), http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS//Vol-159/paper4.pdf.
22. K. Czarnecki and S. Helsen, "Classification of Model Transformation Approaches," *OOPSLA Workshop on Generative Techniques in the Context of Model-Driven Architecture*, Anaheim, CA (2003), http://www.lcc.uma.es/~av/MDD-MDA/MdaEstandares/P7_czarnecki_helsen.pdf.
23. W. M. P. van der Aalst, A. H. M. ter Hofstede, and M. Weske, "Business Process Management: A Survey," *Proceedings of the 1st International Conference on Business Process Management,* Eindhoven, The Netherlands (2003), pp. 1–12.
24. G. Piccinelli and S. L. Williams, "Workflow: A Language for Composing Web Services," *Proceedings of the 1st International Conference on Business Process Management*, Eindhoven, The Netherlands (2003), http://www.cs.iastate.edu/~lumpe/WCL2002/Camera/Piccinelli.pdf.
25. K. M. van Hee, N. Sidorova, L. Somers, and M. Voorhoeve, "Consistency in Model Integration," *Proceedings of the 2nd International Conference on Business Process Management,* Potsdam, Germany (2004), pp. 1–16.
26. R. Anzböck and S. Dustdar, "Semi-Automatic Generation of Web Services and BPEL Processes—A Model-Driven Approach," *Proceedings of the 3rd International Conference on Business Process Management,* Nancy, France (2005), pp. 64–79.
27. A. Gerber, M. Lawley, K. Raymond, J. Steel, and A. Wood, "Transformation: The Missing Link of MDA," *Proceedings of the 1st International Conference on Graph Transformation,* Barcelona, Spain (2002), pp. 90–105.
28. K. Duddy, A. Gerber, M. Lawley, K. Raymond, and J. Steel, "Model Transformation: A Declarative, Reusable Patterns Approach," *Proceedings of the 7th IEEE International Enterprise Distributed Object Computing Conference*, Brisbane, Australia (2003), pp. 174.
29. M. Peltier, "MTrans, a DSL for Model Transformation," *Proceedings of the 6th IEEE International Enterprise Distributed Object Computing Conference,* Lausanne, Switzerland (2002), pp. 190–199.
30. A. Borgida and L. Serafini, "Distributed Description Logics: Assimilating Information from Peer Sources," *Journal of Data Semantics* **2800**, No. 1, pp. 153–184 (2003).
31. F. Budinsky, D. Steinberg, E. Merks, R. Ellersick, and T. J. Grose, *Eclipse Modeling Framework (The Eclipse Series)*, First Edition, Addison-Wesley Professional, Boston, MA (2003).
32. S. Kumaran and P. Nandi, "Adaptive Business Objects: A New Component Model for Business Applications," *Proceedings of the 7th International Conference on Enterprise Information Systems*, Miami, FL (2005), http://www.research.ibm.com/people/p/prabir/ABO.pdf.
33. A. Nigam and N. S. Caswell, "Business Artifacts: An Approach to Operational Specification," *IBM Systems Journal* **42**, No. 3, 428–445 (2003).
34. E. F. Codd, S. B. Codd, and C. T. Salley, *Providing OLAP (On-Line Analytical Processing) to User-Analysts: An IT Mandate* Technical Report, E. F. Codd & Associates Sunnyvale, CA 94085 (1993).
35. R. Kimball, L. Reeves, M. Ross, and W. Thornthwaite, *The Data Warehouse Lifecycle Toolkit*, John Wiley & Sons, New York (1998).
36. C. Baragoin, G. Balasubramaniam, B. Chandrasekharan, L. DelSordo, J. B. Lillelund, J. Maw, A. Neroda, P. Pereira, and J. A. Ramos, *DB2 Cube Views: A Primer*, IBM Redbooks (2003), http://www.redbooks.ibm.com/redbooks/pdfs/sg247002.pdf.
37. L. Zeng, H. Lei, M. Dikun, H. Chang, K. Bhaskaran, and J. Frank, "Model-Driven Business Performance Management," *Proceedings of the IEEE International Conference on e-Business Engineering,* Beijing, China (2005), pp. 295–304.

Accepted for publication December 14, 2004.
Published online July 25, 2006.

Pawan Chowdhary
IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (chowdhar@us.ibm.com). Mr. Chowdhary is an advisory software engineer in the Analytic Models and Architecture department. He is working on the sense-and-respond/BPM architecture framework. He received a B.S. degree in electronics engineering from Nagpur University, India. Mr. Chowdhary is actively involved in the area of the MDD warehouse.

Kumar Bhaskaran
IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (bha@us.ibm.com). Dr. Bhaskaran is a senior manager leading research in the area of service-oriented computing technologies applied to business transformation, business integration, and BPM solutions. He has a Ph.D. degree in engineering science from the Rensselaer Polytechnic Institute.

Nathan S. Caswell
IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (ncaswell@us.ibm.com). Dr. Caswell is a research staff member in the Business Informatics department. He joined IBM Research after earning a Ph.D. in physics from the University of Chicago and holds an IBM Fellowship at the University of California-Berkeley. His recent work has involved developing formal models of business operational behavior. Dr. Caswell holds several patents, has received project-related awards, and has authored a variety of journal articles.

Henry Chang
IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (hychang@us.ibm.com). Dr. Chang is a Senior Technical Staff Member and a research manager in the Business Informatics department. He leads the research effort in the business performance monitoring and management framework with impact on IBM business integration products. He received a B.S. degree in electrical engineering from National Taiwan University, and M.S. and Ph.D. degrees in computer science from the University of Wisconsin at Madison. He received an IBM Innovation Award for his work on business-to-business collaboration solutions. Dr. Chang is a member of the ACM and IEEE.

Tian Chao
IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (tian@us.ibm.com). Ms. Chao is a senior software engineer in the Business Informatics department. She has a B.A. degree from National Taiwan University and an M.S. degree in computer science from Virginia Polytechnic Institute. Her research interests include BPM using an MDD approach, business collaboration, Web services, and security for business processes. Ms. Chao has received an IBM Invention Achievement Award, holds several patents, and has published papers in many journals and conferences.

Shyh-kwei Chen
IBM Research Division, Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598 (skchen@us.ibm.com). Dr. Chen is a research staff member. His current research interests include XML, model transformation and synchronization, data engineering, and compilers. He received a B.S. degree in computer science and information engineering from National Taiwan University, an M.S. degree in computer science from the University of Minnesota, and a Ph.D. degree in computer science from the University of Illinois at Champaign-Urbana. Dr. Chen is a member of the ACM and IEEE.

Michael Dikun
IBM Business Consulting Services, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (mdikun@us.ibm.com). Mr. Dikun is a software engineer. His interests include BPM, MDD, server-side enterprise application development, and Web application development. Mr. Dikun has B.S. and M.S. degrees in computer science from Iona College.

Hui Lei
IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (hlei@us.ibm.com). Dr. Lei is a research staff member. He works in the areas of e-business and pervasive computing, with a focus on software infrastructure and data management issues. He has a Ph.D. degree in computer science from Columbia University. Dr. Lei is a Senior Member of the IEEE.

Jun-Jang (JJ) Jeng
IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (jjjeng@us.ibm.com). Dr. Jeng has a Ph.D. degree in computer science from Michigan State University. His interests include BPM, policy-based management, MDD, agent technologies, and formal disciplines of system and software engineering. Dr. Jeng is a member of the ACM and the IEEE.

Shubir Kapoor
IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (shubirk@us.ibm.com). Mr. Kapoor is an advisory engineer in the Analytic Models and Architecture department. He received an M.S. degree in computer science from Pune University, India. His technical interests include service-oriented architectures, server-side enterprise applications, rule-based expert systems, Web-based application development, and diagnostic systems for use in supply chain and e-business applications.

Christian A. Lang
IBM Research Division, Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598 (langc@us.ibm.com). Dr. Lang is a research staff member in the Business Informatics department. He is currently involved in several projects dealing with the scalability aspects of database-management and business-process-monitoring systems. He received an M.S. degree from the Munich University of Technology and a Ph.D. degree from the University of California at Santa Barbara, both in computer science. Dr. Lang is a member of the ACM and the IEEE.

George Mihaila
IBM Research Division, Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598 (mihaila@us.ibm.com). Dr. Mihaila is a research staff member. He has a B.S. degree from the University of Bucharest and M.S. and Ph.D. degrees from the University of Toronto, all in computer science. He also holds an adjunct faculty appointment at Columbia University. Dr. Mihaila's research interests include Web query languages, Web-based information discovery, data integration, data warehousing, event processing, and XML storage and processing.

Ioana Stanoi
IBM Research Division, Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598 (irs@us.ibm.com). Dr. Stanoi is a research staff member in the Intelligent Information Management department. She received a B.S. degree in computer science, a B.A. degree in physics, and a Ph.D. in computer science, all from the University of California at Santa Barbara. Her patents and publications cover exact and approximate query processing, index optimization, XML, publish/subscribe systems, mobile clients, e-commerce applications, and semantics. She has served on a number of conference program committees and is one of the initiators of the Greater New York Area Database/Information Retrieval Workshop.

Liangzhao Zeng
IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (lzeng@us.ibm.com). Dr. Zeng is a researcher in the Business Informatics department. He received a Ph.D. degree in computer science from the University of New South Wales. His research interests are in the areas of Web services, business process and performance management, event-driven systems, and data stream management. ■