Using a model-driven transformational approach and service-oriented architecture for service delivery management

S. Kumaran
P. Bishop
T. Chao
P. Dhoolia
P. Jain
R. Jaluka
H. Ludwig
A. Moyer
A. Nigam

IT (information technology) service providers often assume that efficient and effective service delivery can be achieved by migrating to a standard set of tools. This assumption is true only if the service provider has monolithic control over the scope and architecture of the customer environment. The trend, however, is toward selective outsourcing, customer control over the architecture of IT solutions, and retention of legacy tools. Target environments are extremely heterogeneous, and the ability of the service provider to control them is diminishing. Consequently, there is a need for a new approach to IT service workflow automation and a new generation of service-delivery management systems that support heterogeneity and collaboration. This paper introduces a new approach to automating complex and variable workflows, applies this approach to IT service delivery management (SDM), presents an SDM architecture based on this approach, and discusses an SDM implementation driven by this architecture. Our implementation architecture leverages service-oriented architecture (SOA) principles by defining loosely coupled service components and a service fulfillment pattern that dynamically integrates them. We discuss the modeling of performance metrics for service delivery and describe how the monitoring and management of key performance indicators (KPIs) are supported as an integral part of our SDM platform.

INTRODUCTION

The primary factor that drives companies to outsource services is their desire to focus on core competencies. The challenge for service providers is that the companies expect the same services at a reduced cost without compromising quality. The

problem is compounded by the fact that the service providers often inherit business and IT processes from their customers. Many of the steps in the processes are necessary for compliance and regulatory reasons. Any noncompliance due to oversight or negligence on the part of a service provider can severely impact the image of the company or its finances. It is therefore critical that service providers have the tools and technologies required to monitor and manage the process of service delivery.

A fundamental challenge in service delivery management (SDM) results from the natural tendency of service providers to organize and optimize based on categories of functional expertise. Processes, metrics, execution, and business cases tend to align with these categories. Further optimization creates centers of competency (COCs) based on functional expertise, and these COCs are placed wherever the skills and financial conditions are the most favorable. Creating solutions and services that span these functions and locations becomes increasingly complex. The business challenge for the provider is to manage this complexity among the categories (known as "verticals") and deliver a financially viable service offering.

The solution approach presented in this paper is based on a full life-cycle business-to-technology method developed by IBM Research called model-driven business transformation (MDBT). MDBT is both a business transformation methodology and a set of innovative technologies that allow business strategies to be realized by choreographing workflow tools and human activities.[1]

At the heart of MDBT is a multilayer modeling framework that spans the business and IT domains. The framework is made up of four layers: business strategy, business operations, solution composition, and IT implementation. Each layer constitutes a different level of abstraction, performs a well-defined function, and has a different audience. The *strategy layer* defines the goals and objectives of the business system. The *operation layer* describes the operations performed by the business system to achieve the goals. The *composition layer* is an abstraction of the computational elements that are needed to execute the business operations. The *implementation layer* specifies how the computational elements are implemented on a specific IT platform. In this paper, we describe how this framework helps to effectively address the SDM (service delivery management) challenge.

In the following section, we describe the MDBT approach and the underlying framework. We then define SDM and advocate an SDM platform that supports service delivery operations. Next, we discuss in detail how an SDM platform is realized using the MDBT approach, and we present our experience in employing an SDM platform in real-world service delivery. We conclude with a discussion of related work in this area and an analysis of the innovations introduced in this paper.

Model-driven business transformation

MDBT uses formal models to explicitly define the structure and behavior of a business component, employs these models to monitor, analyze, and improve its performance, and leverages these models in the construction of its IT systems. In the context of the work discussed in this paper, a *business component* is an outsourcing organization which provides IT outsourcing services to clients (hereafter called *outsourcers*). The outsourcer may in turn subcontract parts of the work to specialized service provider organizations.

In the MDBT approach, the transformation process begins with the identification of the strategic goals and objectives of the business component. This leads to a set of initiatives that support these goals. These initiatives determine the definition, analysis, optimization, and implementation of the business operations of the organization such that the strategic goals can be achieved. The output of this step is the definition of a "balanced scorecard" that enables business owners to monitor progress toward attaining the strategic goals. In the service delivery domain, the output of this step is the balanced scorecard for the outsourcer.

Formal definition of the business operations and the operational KPIs (key performance indicators) is the next step in the transformation process. We refer to this as the *business operation model*. For the service delivery domain, such a model formally describes how services are defined and managed, how service requests are created and provisioned, and how service provisioning interacts with change management, release management, and configuration management.

A business operation model is different from the more familiar workflow models. At its core, a workflow model defines the sequencing of activities in a business process and the flow of data through these activities. Thus, a workflow model consists of a control flow graph and a data flow graph. A business operation model, on the other hand, defines the key business artifacts and the operations performed on these artifacts.[2] An example of a business artifact in the service delivery domain is a service order. Operation modeling differs from workflow modeling in the context of business processes much like the object-oriented paradigm differs from the procedural paradigm for computer programming.

In addition to modeling the operations on the business artifacts, the business operation model includes elements to manage business performance as well. This is accomplished by specifying KPIs associated with the artifacts. KPIs are quantifiable measurements, agreed to in advance, that reflect the factors critical to the success of a business. Typically, an organization uses KPIs to manage business performance by measuring progress toward the business goals. In addition to KPI definitions, the business performance model includes relationships between the KPIs, algorithms and data for computing them, business situations that are triggered by them, and remedial actions which may be needed.

KPIs are organized based on the business artifact to which they pertain. Each KPI is qualified by one or more dimensions. A *dimension* is a grouping criterion used to view a KPI. For example, a service-order-volume KPI can be viewed based on time interval, service delivery organizations, accounts, clients, and so forth.

There are several reasons for creating a formal operation model for service delivery. Whereas workflow models may lead to local optimizations by improving a step in the process, operation models can lead to global optimizations by fundamentally changing the way a business operates. A well-known example is how Dell, Inc. became the prominent company in the personal computer business by using business models made possible by the innovations it introduced in the management of its supply chain.[3]

A business operation model can be used to create workflows on demand at runtime without the need for explicit modeling and development, providing the flexibility that is critical in the strategic outsourcing (SO) business. For example, the specific workflows needed for provisioning a service request depend on several factors, including the customer who is requesting the service, the type of service being requested, information contained in this instance of the service request, the current status of service providers, and the service provisioning environment. It is not practical to define all instances of such workflows. Rather, a well-designed business operation model can be used to dynamically create such workflows on demand.[1]

Unlike traditional workflows, which focus on the flow of work items, the business operation model focuses on the flow of information in the business process, resulting in a clear definition of how information is defined, created, used, and manipulated by the business. This focus on information leads to better integration of applications and tools and provides opportunities for exploitation of business intelligence. This differs from activity-based approaches to business intelligence such as the work discussed in Reference 4. The advantages of focusing on the artifact as opposed to the activity are discussed in detail in Reference 5.

The next step in MDBT is the judicious use of technology to support the execution of business operations. This involves the generation of a platform-independent solution composition model and the realization of this model on a specific software platform. By software platform, we mean a set of middleware products or applications. The WebSphere* Process Server[6] is an example of a software platform for service delivery.

There are several benefits to having a platform-independent solution composition model. Model-driven code generation can be used to realize the solution on a specific software platform, leading to significant reduction in time to value. By using a different code generation plug-in, the same model can be used to realize the IT solution on a different platform. This capability makes it considerably easier to support changes in the software platform. The business-critical elements of the IT solution can be generated automatically from the business operation model, effectively bridging the business-

IT gap. The generated model can then be enriched with the IT-level details that are important for creating a full-fledged solution.

The platform-independent *solution composition model* is the formal description of a computer program that implements the functionality required by the business operation model. In other words, the business operation model formalizes the business requirements, whereas the solution composition model formalizes the solution design. The key elements in this model are the service choreography components referred to as adaptive business objects[7] (ABOs). The computational model of an ABO is a communicating finite state machine (FSM) that captures the life cycle of a business artifact from creation to archiving. The transformation from the operation layer to the implementation layer leads to the automatic generation of FSMs. In addition to the FSM, an ABO model includes client interfaces and the complete data model of the business artifact.

Once the FSMs are generated, the solution composition model allows specification of actions associated with state transitions in the FSMs. An action implements the *command design pattern*[8] and can be bound to the execution of one or more services based on service-oriented architecture (SOA). By using the FSMs and the actions associated with state transitions, the solution composition model can define the orchestration of one or more SOA-based services. Thus the solution composition model leverages the principles of service-oriented architecture (SOA) by composing reusable, stateless services to create composite applications that realize the new functionality demanded by the business operation model. This leads to a modular, component-based, distributed, and scalable solution architecture.

In addition to ABOs, the solution composition model includes elements for business performance management (BPM) as well. Included in BPM is a *monitoring model* for monitoring runtime business process performance and a *visibility model* to enable "dashboard" (i.e., at-a-glance multiple display) views and the capability to analyze results in detail. These models are manually created from the KPI models at the business operation level.

The next step is to create an implementation of the IT solution on a specific IT platform. Currently, we generate code for the IBM WebSphere platform and DB2* database from the platform-independent solution composition model. Once the solution is deployed, business owners can monitor and analyze business performance using KPIs and continuously improve the models, both at the business and IT levels, based on this performance analysis. A BPM dashboard is used to monitor and analyze KPIs and to deliver real-time alerts to stakeholders. The BPM dashboard employs an online analytical processing (OLAP) component to extract real-time data generated by the IT solution artifacts to enable employees to make informed decisions and to respond quickly and proactively to business opportunities and threats.

Figure 1 shows the MDBT framework, including the separation of concerns, connections between model layers, and the closed-loop architecture achieved using the BPM component. For example, the business operations layer is concerned with the understanding of the business by business owners. It defines the business operation model and observation model to support business services and strategic KPIs defined in the layer above. At this layer, the expectations for the BPM component are to determine if business commitments are being met. This determination is made based on the organization's response to business situations. Information compiled in this layer can be used to optimize the business operations.

Before discussing the details of the SDM platform derived by using the MDBT approach, we define IT service delivery management in the following section and advocate the use of an SDM platform in managing the service delivery processes.

Service delivery management

IT service delivery management focuses primarily on the running of an IT service delivery business, irrespective of whether the services are being delivered by in-house IT teams or are outsourced. IT service delivery spans many operational processes, such as incident management, problem management, change management, configuration management, availability, service request management, and service support. Like any business, the service delivery organization needs to focus on who its customers are, what services they need, and what delivery capabilities are necessary to deliver those services. It is therefore important not only to build a

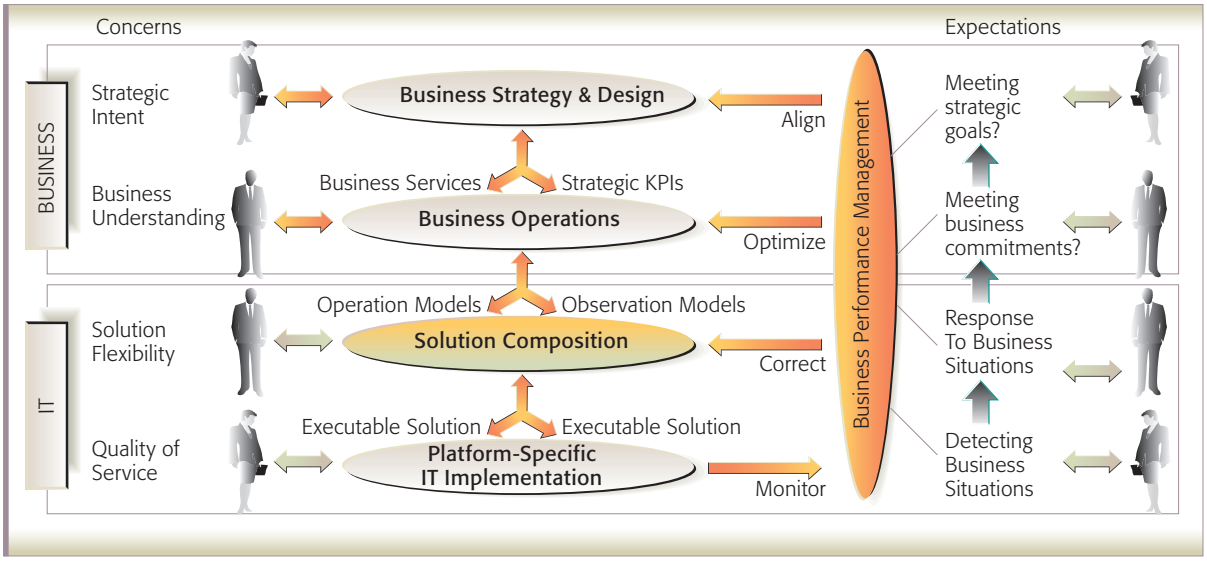

Figure 1
MDBT framework

catalog of services but to link these services to delivery capabilities. This allows the service delivery organization to make services more granular and to develop more value-added services from existing delivery capabilities.

An IT delivery business can be organized by geographic area, by competencies, or by a combination of both. Delivery teams can be local (serving local customers) or global (serving any customer). Teams can be comprised of generalists (with one person performing end-to-end tasks to fulfill a service) or specialists (with highly specialized team members performing well-defined tasks). A service delivery organization needs a service delivery platform that can support any of these organizational models to orchestrate and manage the fulfillment of services irrespective of the tools and technologies used by the different delivery teams in various parts of the organization. In the following sections, we discuss the details of an SDM platform that was designed and implemented by using the MDBT approach.

Service delivery business and operational goals

Businesses can use KPIs to measure progress toward the organizational goals they have defined. Service providers can also use KPIs to focus on the service delivery goals that differentiate them from their competitors.

Among the generic business goals that apply to both profit-center and cost-center business models of service delivery are the providing of services that customers value, achieving the lowest cost in providing the requisite value at the negotiated price, achieving the highest quality consistent with the level of value, and achieving high customer satisfaction.

At the heart of all service delivery businesses is the struggle to strike a delicate balance between delivering contracted services at the lowest price and at the level of value that was agreed upon during contract negotiations and avoiding non-negotiated value that drives up costs unnecessarily. SDM systems must therefore capture the true costs of offered services at various value levels on an ongoing basis so that those costs can be used by marketing and sales personnel when negotiating service contracts. In addition, SDM systems must capture non-cost data that affects the customer's experience, such as time to delivery, quality of service, and the flexibility to meet new demands.

Once the business goals have been established for each phase of the life cycle, the service delivery organization must apply SDM practices to the daily operations of each phase. This leads to the formulation of operational goals, which in turn provide guidance for designing the overall operation

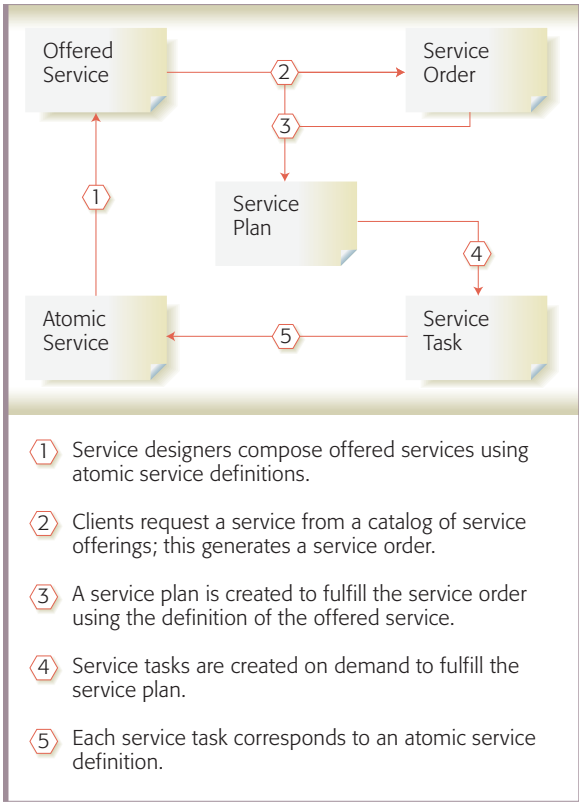

Figure 2
SDM business artifacts

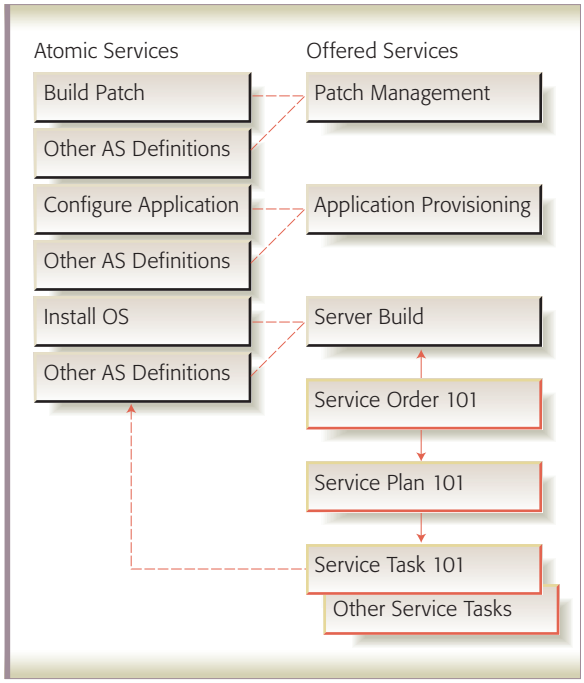

Figure 3
Examples of artifact instances

of the business. As we describe later in the paper, this operation consists of processing certain business artifacts; specifically, service requests, service orders, and service tasks.

AN OPERATIONAL MODEL OF SDM

The fundamental abstractions of MDBT are business artifacts that are processed by business tasks and stored in repositories. There are five types of business artifact in SDM as listed below. ***Figure 2*** shows these artifact types and the relationships among them. ***Figure 3*** shows examples of artifact instances.

1. *Atomic service*—This is a reusable service element that reflects a specific capability. It is a unit of work that has well-defined boundaries and can be performed by a service provider. Any further decomposition of this service element is not meaningful to the service delivery business. Examples of atomic services include building a software patch, installing an operating system, and configuring an application. An atomic service artifact contains a definition of the service. This definition includes the specification of the input data required to perform the service and the output data generated after successful completion of the service. Further details on the atomic-service artifact, including its attributes and task structure, can be found in Reference 9.
2. *Offered service*—A set of atomic services can be composed into an offered (or composite) service. Examples of offered services include "server build" (i.e., configuring and provisioning a server), patch management, and application provisioning. An offered-service artifact contains a definition of the offered service. This definition includes the specification of the input data required to perform the service, the output data generated after successful completion of the service, the atomic services required to perform the offered service, and sequencing constraints on the atomic services. Further details on the offered-service artifact, including its attributes and task structure, can be found in Reference 9.
3. *Service order*—Whereas atomic-service and offered-service artifacts are used to configure an SDM system, service orders, service plans, and service tasks are used for service fulfillment. A service order represents a request from a client

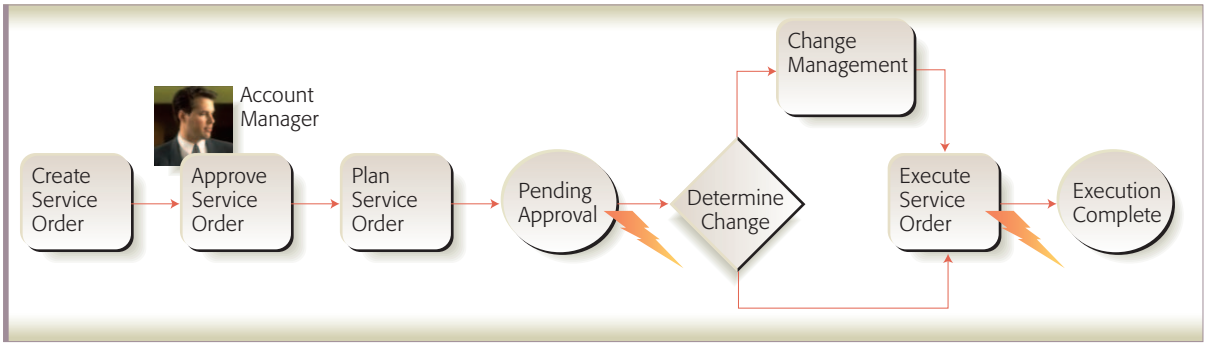

Figure 4
Life cycle of service-order business artifact

for a service. Clients create service orders using a service catalog. A service catalog is a projection of a subset of offered services for the client domain. For example, a client may request a server build, resulting in a service order. The service order captures both the generic and the offered-service-specific attributes of the service request. Generic attributes include the order identifier, description, priority, planned time, planned and actual start and end times, status, and service-level agreements (SLAs). Service-specific attributes may follow a canonical data element schema that has been published and accepted as a standard. The task structure of a service order is modeled at a level of granularity that permits standardization across accounts and service types while allowing flexibility to adapt to specific needs of an account or service type.

Figure 4 shows the life cycle of the service-order artifact. The essential business tasks that act on the artifact are:

a. *Create service order*—This task can be performed manually through the user interface of the SDM or programmatically by sending a request to the service request gateway of the service-order artifact.
b. *Approve service order*—The account manager performs entitlement checks on the service request before approving the request.
c. *Plan service order*—In this task, the planning needed for executing the service order is performed.
d. *Manage change*—If the service order requires change approval, change requests are created as appropriate. This task is completed when the change approval is obtained.
e. *Execute service order*—This task begins when the work needed to execute the service order begins and is completed when this work is performed.

4. *Service plan*—Using the offered-service definition of the service composition corresponding to the service order, a service-plan artifact is created. Whereas the composition of the atomic services in the offered-service definition can be thought of as an abstract plan, the service plan is a concrete plan generated from the abstract plan but customized for the specific instance of a service order. For example, a service plan may be created to fulfill a specific service order. It allows runtime overrides of the generic plan associated with offered services, thus leading to a greatly reduced generic set of offered-service definitions. ***Figure 5*** shows the life cycle of the service plan artifact. The business tasks that operate on the service plan artifact are as follows:

a. *Create service plan*—A planning service performs this task based on the service plan template associated with the corresponding offered service. The planning service applies rules and policies to perform context-based overrides on the abstract plan. It then applies the provider assignment policies (weighted functions of cost, time, default provider linkage, and workload manage-

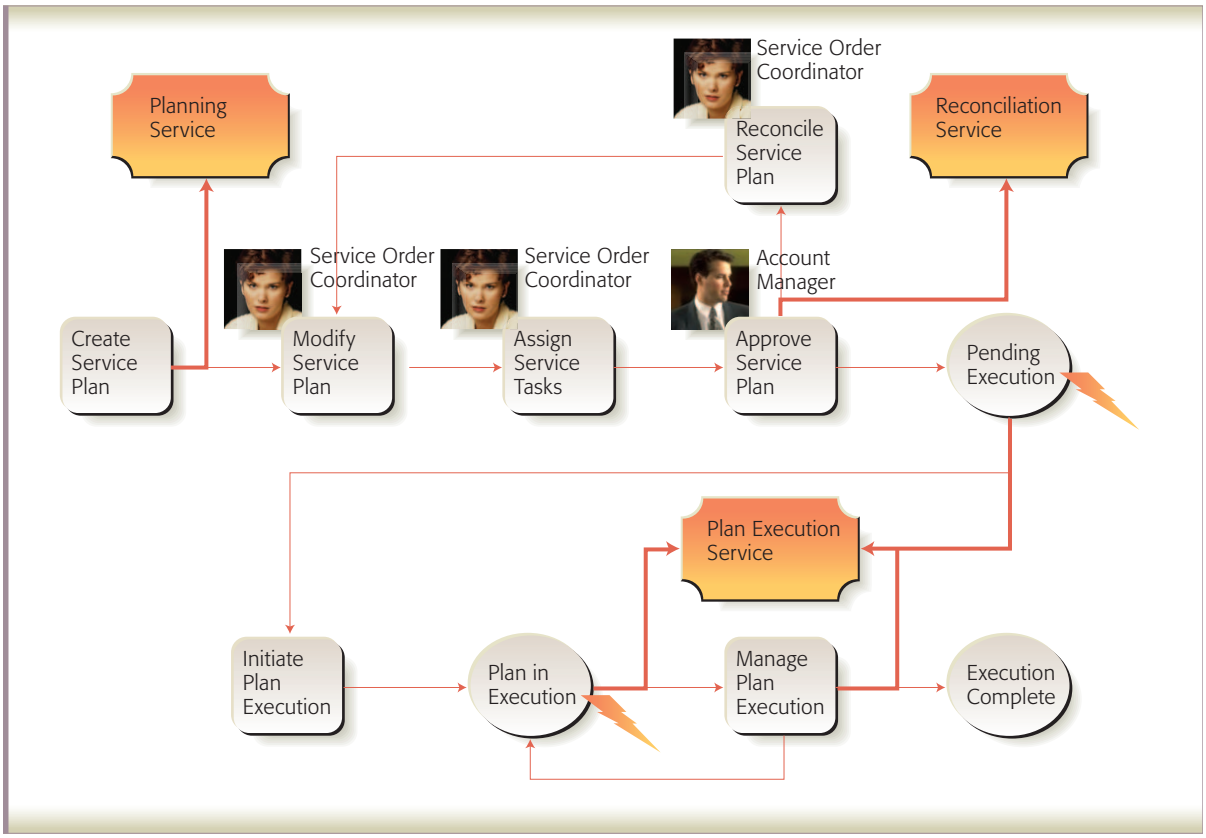

Figure 5
Life cycle of service-plan business artifact

ment) to perform automated provider assignments.

b. *Modify service plan*—This task is performed as needed while the plan is in execution to respond to unexpected events.
c. *Assign service providers*—Default assignments by the planning service can subsequently be overridden manually.
d. *Approve service plan*—All stakeholders need to approve the plan before it can begin execution.
e. *Initiate plan execution*—This task initiates the execution of the atomic services contained in the plan.
f. *Manage plan execution*—This task is triggered every time an atomic service is completed. It updates the plan status and triggers the execution of the next set of atomic services as needed.

5. *Service task*—One or more service-task artifacts are created based on the service plan. Each such service-task artifact corresponds to an atomic-service definition (ASD). The service-task artifact includes instance data needed to perform the corresponding atomic service. Service tasks are routed to service providers to perform the work encapsulated in the ASDs. For example, a service task may be created to install an operating system as part of fulfilling a given service order under a given service plan. ***Figure 6*** shows the life cycle of the service-task artifact.

Next, we discuss the business performance model for SDM. ***Table 1*** and ***Table 2*** show the KPIs and associated dimensions for the service-order and service-task artifacts. Management policies are defined by using the notion of a "situation." A situation is detected by applying a situation rule to one or more KPIs. Management policy defines the actions to be taken if a situation arises. For example, a management policy may specify that the elapsed time of a service order should not exceed a certain threshold, as defined at the SLA level or at an

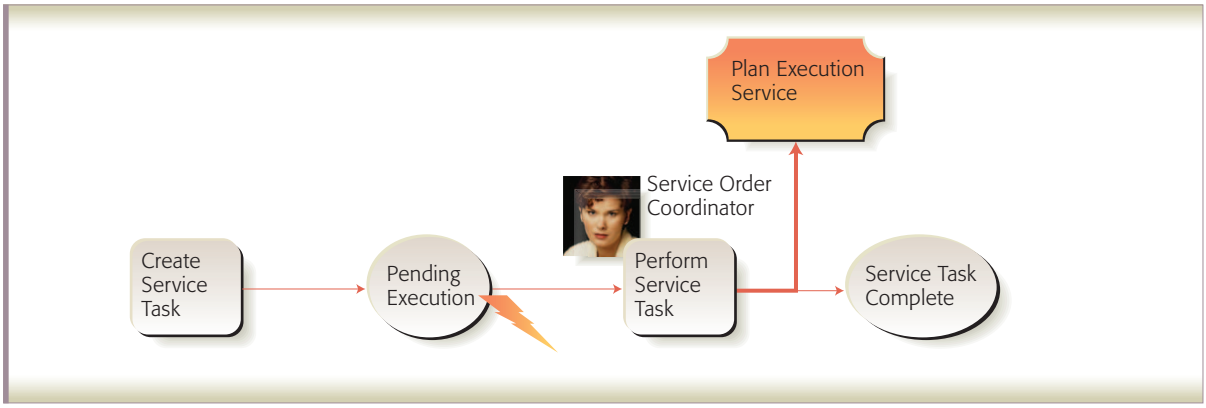

Figure 6
Life cycle of service-task business artifact

individual service order level. In this case, the policy may be defined by specifying the situation name as "Elapsed time exceeds threshold," the KPI as service order age, and the evaluation frequency as continuous. The situation rule would be, "Trigger the situation when elapsed time is 90 percent of the planned time and the service order is not completed," and the action would be to notify stakeholders of the situation.

Business artifacts logically partition the business operations space into orthogonal subspaces suitable for flexible and loosely coupled solution composition. Each subspace contains the task structure of an artifact.[10] These task structures show the operations being performed on each artifact, the business role that performs each operation, the repositories used by each artifact, and the dependencies between them. The operation model for SDM is the union of these task structures. Interactions between the artifacts define the boundaries of these subspaces. These interactions are captured as a set of design rules.[10] Independent processing and tooling innovations can then be applied to these subspaces, as long as the design rules governing their interaction are obeyed.

Table 1 Examples of KPIs and dimensions for the service order artifact

KPI	Applicable Dimensions
Volume	Time interval, service type, account, organization
Average turnaround time	Time interval, service type, account, organization
Percentage late	Point in time, service type, account, organization
Average age	Time interval, point in time, service type, account, organization

Table 2 Examples of KPIs and dimensions for the service task artifact

KPI	Applicable Dimensions
Average time to perform	Time interval, organization, service type
Volume	Time interval, organization, service type

Platform-independent solution composition model

The core elements of the solution composition model are generated programmatically from the business operation model, and the model is then manually extended. The core elements of the solution composition model are: ABOs, which represent business artifacts at the IT level; an information view model that describes the client-side representations of the ABOs; a use-case realization model that describes how users perform business tasks by invoking operations on the ABOs; a service integration model that describes how legacy services are consumed by the ABOs as part of completing business tasks; and a BPM model. These elements are described in detail in the following

subsections. The details of the algorithm for transforming the business operation model to the solution composition model can be found in Reference 11.

ABOs

In the solution composition model, there are five ABOs that correspond to the five business artifacts in the business operation model; namely, the atomic-service, offered-service, service-plan, service-order, and service-task artifacts. An ABO model contains structural properties that describe the data structure of the ABO and its associations with other ABOs and business objects, operations that describe the interface by which clients communicate with an ABO (these include the call triggers that can be used to pass events asynchronously to the ABO), and a life-cycle model. The life-cycle model includes the states of the ABO corresponding to life-cycle stages of the underlying business artifact, state events that determine when the ABO enters or leaves a state, transitions that represent the act of moving from one state in the life cycle to another, and actions that are invoked as part of the transitions. These actions include data actions that manipulate ABO data, invocations of legacy services, and publishing of events to other ABOs.

ABO models are created automatically by transforming the business operation model. *Figure 7* shows a partial view of a service order ABO.

Information view model

List views and state views are the core elements of the information view model. These elements define access constraints on the information presented to clients by the ABOs. These views are defined for every valid combination of role and artifact type. There are three constraints on the list views. The *search* constraint defines the business contexts in which the artifacts are searchable by a user playing a specific business role. The *state* constraint defines the states in which the artifacts are visible to users playing specific business roles. The *summary data* constraint defines the set of attributes that a user playing a specific business role can see in a summary view of the artifact.

List views are used to create a list of artifact instances and the summary data describing them. A user can then select one of the artifact instances from the list to perform a business task. This results in the creation of an artifact representation which enables the user to perform the task.

The state view is used to model the artifact representations. There are two constraints on the state view. The *execute* constraint specifies what operations can be performed by a user in a specific role on an artifact type in a given state. The *write* constraint specifies what subset of the artifact data model is writable for an artifact type in the given state.

A default set of list views and state views is created automatically by transforming the business operation model. Additional views are defined manually as needed.

Use-case realization model

Two types of use-case realization models are used for solution composition. The models of the first type are created corresponding to each business task in the business operation model. Each such model links a set of business roles, a set of list views and state views, and a set of ABOs for the purpose of performing a business task. For example, there is a use-case realization model to specify that the "SO coordinator" performs the assigning-service-providers task through specific views of the service-plan artifact. Another use-case realization model specifies that the account manager performs the account-manager-approval task through the corresponding information views of the service-order artifact.

The other type of use-case realization model defines the navigation scenarios for each business role. For example, there is a use-case realization model to specify how users playing the account manager role navigate through service orders and service tasks that pertain to them through the list views of the service-order and service-task artifacts.

A default set of use-case realization models is created automatically by transforming the business operation model. Additional models are defined manually as needed.

Service integration model

This model is used to define the binding of services invoked as part of the state transitions of an ABO. For the SDM platform, these include the planning service, scheduling service, plan-reconciliation service, and plan-execution service. For example, the plan-execution service binds to a specific state transition of the

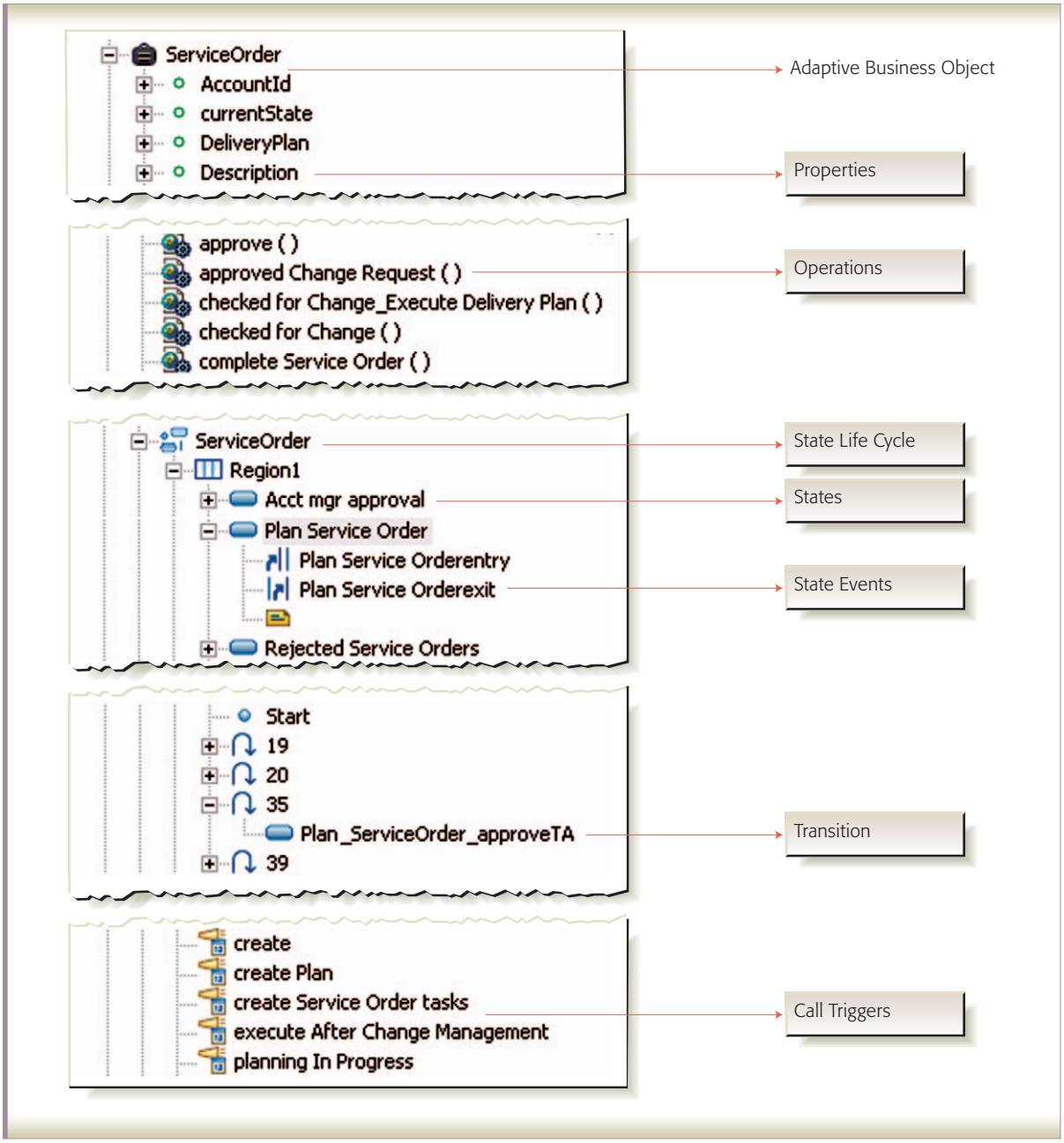

Figure 7
Typical artifact of platform-independent model

service plan ABO. The service integration model is added manually to the solution composition model.

BPM model

A BPM model is added manually to the solution composition model based on the KPI, situation, and action specifications at the business operation level. This model includes elements for monitoring run-time business process performance, the visibility model for dashboard views, and the capability to analyze results in detail.

SOLUTION IMPLEMENTATION

An SDM implementation on a specific platform can be generated from the platform-independent solution composition model through model-driven code

generation. Using this approach, we implemented the SDM solution on two IBM platforms: (1) the J2EE** platform, WebSphere Business Integration Server Foundation,[12] and (2) the SCA (Service Component Architecture) platform, WebSphere Process Server.[6]

The implementation on the WebSphere Process Server is made up of SCA components for business process choreography, user interface components, and BPM components. These components are discussed in detail in this section.

SCA components

The key element types of the SCA-based component assembly are artifact services, business state machines, service components, and service data objects (SDOs).

Artifact services are the service interfaces of ABOs. They enable the interface to access or modify artifact data, perform synchronous operations on the artifact, and receive information about external events that are significant from the perspective of the artifact life cycle. Artifact services are implemented as stateless session beans. *Business state machines* are used to manage the life cycle of an artifact.

Service components represent the services that are invoked by ABOs as part of state transitions. Key service components are the planning service (used to create a concrete service plan for the execution of the service order), the change request creation service (which determines if a change request needs to be created for the service order), and the plan execution service, which uses the service plan to determine dependencies between the service tasks and then programmatically initiates the execution of these tasks by communicating with the respective service providers. The plan execution service orchestrates the complete execution of the service order by ensuring that all service tasks are executed as specified in the service plan.

Service data objects[13] represent the primary client programming model offered by the SCA-based implementation. Clients receive SDOs when they invoke an artifact service for retrieving artifact data. Similarly, clients post SDOs back to the SDM platform to request processing of modifications to artifact data.

User interface

The user interface of the SDM platform is generated and extended by using the MDBT method. Through model transformation, a simple user interface is first generated automatically based on a hidden Markov model of user interaction.[14] In this interaction, the user goes through system authentication; the system presents a list of artifact instances pending user action; the user selects a particular artifact instance; the system renders a detailed representation of the selected artifact instance with relevant artifact details, related business items, and applicable processing options; the user selects a processing option, optionally adds or modifies data, and submits the form for processing; and the server processes the request and returns the status to the client.

The generated user interface is manually enhanced for "look and feel."

BPM

Platform-independent models of BPM are transformed to generate runtime code for monitoring and managing business performance. This includes code for event correlation and aggregation, KPI evaluation, situation detection, dashboard rendering, and data warehouse schema; it also includes ETL (Extract, Transform, and Load) software to extract data for use by OLAP components. In our SDM implementation, IBM Alphablox*[15] provides the OLAP functions.

Use of standards in our SDM solution

In this section, we briefly discuss the use of standards in the design, development, and execution of our SDM solution. UML2[16] (Unified Modeling Language** 2) is used as the modeling language for business operation modeling and platform-independent solution-composition modeling. Our solution implementation leverages J2EE[17] technologies.

The SDM platform uses a set of Web-services-based interfaces for legacy application integration. These interfaces are defined using the WSDL (Web Services Description Language) standard.[18] SDOs are used for communication between various SDM components.[13] WSDL is used to define the service requestor interface for atomic service invocation. Service providers implement this interface. At runtime, an SDM component invokes this interface to pass the information needed to perform the work to the service provider.

We use WS-BPEL[19] (Web Services Business Process Execution Language) to define the composition of atomic services in an offered-service artifact and a service-plan artifact. At runtime, these BPEL definitions execute dynamic and federated workflows by using concurrent communicating state machines.

Deployment experience

This section describes our experience in using the SDM platform for IT service delivery in a real-world setting. The SDM platform was used to provision server build requests and for software patch management. Though the end-to-end process models for these functions appear to be very different, we were able to map both of these processes uniformly to the SDM architecture by using the five SDM artifacts described previously. The offered services were "server build" and "patch management." The server build process included the atomic services "allocate server space," "assign IP address," "build server," "connect to network," "order hardware," "prepare server configuration data," "preproduction configuration," and "ship." The patch management process included the atomic services "patch analysis," "APAR (authorized program analysis report) server mapping," and "patch apply."

Significant improvement in efficiency was achieved by using the SDM solution for service delivery. This resulted from the SDM features of (1) a streamlined business process (eliminating several legacy applications and "non-value-add" steps), (2) workflow automation, (3) catalog-based service request creation, (4) a single SDM portal to manage the end-to-end life cycle of a service order, (5) programmatic integration with legacy applications, (6) automated plan generation based on the composite service plan definition in the catalog, and (7) automated service provider assignments based on atomic service properties.

Asset reuse was improved because SDM processes for managing service orders and service tasks were reused across offerings. There was a continuous improvement of service delivery efficiency through selective optimization of atomic services. Users were able to monitor service provider performance. Resiliency was improved because tasks could be rerouted to a different provider if there were problems with the execution of a specific atomic service.

RELATED WORK

Most of the previous work in linking business processes to IT involves workflow modeling.[20] Workflow management systems reduce costs, assure process control, and increase the quality of both process execution and workflow deliverables.[21] Workflows support a wide variety of business processes. Workflow management systems are complex applications that must marshal both process activities and information flow while assigning work to persons or technologies in a synchronized sequence. Because of the complexities involved and the price factors required to address all possible workflow scenarios, workflow management systems tend to focus on one particular type of workflow management.

Historically, the IT outsourcer has applied workflow management systems to address specific issues in the IT environment or has simply inherited workflow management systems from the customer's previous IT service provider. This typically led to a jumble of workflow management systems, some purchased and others developed in-house. Several attempts have been made to replace these disparate systems with a single enterprise system, but no single workflow engine can address all of the conflicting requirements.

The MDBT approach is fundamentally different from traditional workflow-management system design. Instead of focusing on the process of the workflow, the solution focuses on the state changes in the life cycles of the business artifacts. Each state change may launch an action. That action may be the execution of a procedure or a program which performs that procedure automatically. These actions may be thought of as business tasks that are executed much as they are in traditional workflow management systems. The granularity of a business task requires that it effect a change in the related business artifact. The totality of associated business tasks constitutes the process of the workflow.

Applying the MDBT method and SOA principles results in a unique solution that resolves many conflicts. The resulting object-based application choreographs the execution of tasks either by the application itself or by any number of external workflow engines in a form of workflow federation.

The business goal in service delivery is to maintain a thin layer of centrally controlled management and to allow maximum flexibility in the performance of tasks. Our solution harvests the expertise of the people responsible for the detailed, day-to-day IT service work and allows those workers the latitude to perform their jobs in a way that best suits the business. This is accomplished by restricting the service task ABOs to the execution of an abstraction of a service in the form of an atomic service. The internal operations of the atomic service are not described or mandated; they are left to the service performer to determine. Multiple service providers may be used to perform an atomic service, each following its own best practices or those of a governing body as determined by the business.

Each service provider may also determine the best tooling to assist in its work, as long as the requirements of the ASD are met. Thus, manual or external workflow management automation engines may perform an atomic service as long as they satisfy applicable business controls. Alternatively, the business may mandate that a specific workflow-management automation engine be used. Because the service order and service tasks do not depend on the use of specific tools or manual procedures, the performance tasks are decoupled from the managing tasks, allowing service tasks to be performed anywhere, by anyone, by any means that meets the base specifications of the ASD.

In a seminal *IBM Systems Journal* paper,[22] Zachman proposed a framework for enterprise architecture as a "two dimensional classification scheme for descriptive representations of an enterprise." Though there are similarities between the framework described in this paper and Zachman's work, there are some fundamental differences, most importantly that our framework is not a classification scheme but a formal model of the behavior and structure of an enterprise entity. Hence, the connections between the layers are extremely critical for us. Instead of having disjoint cells representing abstractions in each column of the classification scheme (as pertained in Zachman's work), we use a multilayered model. Each layer (i.e., perspective) has a base model that defines the core structure and behavior of the perspective and enhancements that add more information and are orthogonal to each other.

Through its Model-Driven Architecture** (MDA**) initiative,[23] the Object Management Group (OMG**) is working to create the standards necessary to facilitate a comprehensive new approach to the creation, integration, and maintenance of software assets. MDA models are organized vertically in three layers, with computation-independent models (CIMs) at the top, platform-independent models (PIMs) in the middle, and platform-specific models (PSMs) at the bottom. CIMs are domain models that capture the requirements of the solution. PIMs and PSMs describe the solution structure at different levels of abstraction.

Our framework is thoroughly aligned with the MDA approach. The strategy and operation layers map to the CIM layer of MDA. The computation and implementation layers map to the PIM and PSM layers. We apply the MDA approach to create a model-driven enterprise through precise definition of model elements in each layer and bidirectional links between layers (including algorithmic transformations between the layers). Model-driven architectures are only as good as the models themselves; the hardest problem is identifying the right abstractions (models), which is the focus of our work.

SOA uses services as the basic building blocks for creating software assets.[24] The computation layer in our framework models the service choreography and service-brokering components necessary to create an SOA-based business integration and management solution. Our framework brings MDA and SOA together to create an adaptive enterprise system with business processes as the organizing principle.

Wu et al.[25] discuss issues in using BPEL for Web services composition and propose a synchronization expression language called DSCL (DAG [directed acyclic graph] Synchronization Constraint Language) to specify synchronization constraints in procedural process specification languages. This approach can effectively reduce the development effort for process designers. This work is complementary to our approach, as we can use DSCL in defining the service compositions contained in the offered-service artifact.

Sauvé et al.[26] present a three-layer basic business-driven IT management (BDIM) model in their introductory overview and survey of this field. Their business layer maps to our strategy layer, and

their business process layer maps to our operation layer. They suggest that the contents of the IT service layer in their model can be organized as a collection of sublayers for a cleaner structure, and this is what we have done in creating the platform-independent computation layer and the platform-specific implementation layer. Further, we use the abstraction of a business artifact as a key organizational element in defining the contents of our framework, develop formal models to define the layers, and provide model-driven transformations between layers.

Tosic[27] describes five challenges of BDIM and five approaches to addressing them, all of which are handled holistically by our framework. Brenner[28] presents a taxonomy of ITIL** (Information Technology Infrastructure Library**) processes based on the dimensions of structure and organizational complexity and argues that workflow tools are of limited use in supporting processes that are not very structured. The innovations we introduce at the computational level, specifically the communicating state machine model, make it easier to provide tool support for processes that are not highly structured.

Danciu[29] analyzes formalisms designed for business process representation, assesses their suitability for expressing IT management process definitions, and categorizes the examined formalisms according to IT management requirements. The formalisms we employ are fundamentally different from those discussed in this work and demonstrably superior in their ability to meet IT management requirements. Mayerl et al.[30] analyze service-level management processes to identify requirements for management applications and propose integration of applications using Web services.

CONCLUSION

The work described in this paper has made a number of significant contributions in advancing the state of the art in service delivery. Most important is the design and implementation of a new approach for organizing a service-request processing system, composed of a receiving operational model and a business performance model based on domain knowledge, a solution model that is developed based on this domain knowledge, and a service delivery platform implemented to perform service-request processing.

This approach has several advantages over today's service-delivery management systems. It provides a clear separation of the service delivery platform from the service offerings, thereby leading to dynamic support for new offerings. It enables global-delivery model-based process automation with planning and scheduling components which factor location-based variables into the planning process; it provides an optimized set of service offerings with parameterized and rule-based service plans; it provides a scalable competency-based approach to service delivery; it organizes delivery capability along with optimal competencies based on an on demand model; and it provides a service-delivery performance management platform monitored by KPIs, with support for notification of any violation of SLAs.

The modeling framework presented in this paper adapts and extends workflow technologies to address the unique requirements of the IT service management domain. These extensions are primarily in three areas: life-cycle management, dynamic process execution, and federated workflow.

The artifact concept enables end users to manage workflow definitions as any other data through the user interface of SDM. Thus, there is no distinction between a "runtime" and a "build time" with respect to managing the workflow life cycle.

By defining the service-order, service-plan, and service-task artifacts (and through their collaboration), the system supports highly dynamic process execution scenarios. For example, a plan could be revised mid-stream, new service tasks could be created as necessary, existing service tasks could be canceled, and so on.

Using a monolithic workflow engine for end-to-end service delivery management is highly impractical because service provider organizations typically use their own workflow tools. By defining the service-plan and the service-task artifacts (and through their collaboration), the definition of the work to be performed by service providers is packaged and shipped to them through Web services. Service providers perform this work using their own workflow tools and detailed work-breakdown structures, and send the results back to the service coordinators.

As the functionality of MDBT is deployed, it facilitates higher levels of efficiency for service delivery management. Armed with MDBT-enabled metrics such as skills, costs, and duration of a service, as well as the KPIs of different locations and performers, providers can build a plan that quickly enables their service to take advantage of global efficiencies or react to global conditions, resulting in a higher level of resiliency.

Further MDBT exploitation is expected in the area of baselines. Collecting the metrics and KPIs for the first accounts will establish a baseline for the execution of a service versus the MDBT operational model for that service. That baseline will allow service delivery personnel to more clearly evaluate the impact of customizations that deviate from the operational model of the service.

CITED REFERENCES

1. S. Kumaran, "Model Driven Enterprise," *Proceedings of the Global Enterprise Application Integration Summit*, Banf, Canada (2004), pp. 166–180.
2. A. Nigam and N. S. Caswell, "Business Artifacts: An Approach to Operational Specification," *IBM Systems Journal* **42**, No. 3, 428–445 (2003).
3. M. Hammer, "Deep Change: How Operational Innovation Can Transform Your Company," *Harvard Business Review* (April 2004).
4. M. Castellanos, F. Casati, M.-C. Shan, and U. Dayal, "iBOM: A Platform for Intelligent Business Operation Management," *Proceedings of the 21st International Conference on Data Engineering*, Tokyo, Japan (2005).
5. 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).
6. WebSphere Process Server, IBM Corporation, http://www.ibm.com/software/integration/wps/.
7. S. Kumaran and P. Nandi, "Adaptive Business Objects—A New Component Model for Business Integration," *Proceedings of the Seventh International Conference on Enterprise Information Systems (ICEIS 2005)*, Miami, Florida (2005).
8. E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns—Elements of Reusable Object Oriented Software*, Addison-Wesley Publishing Company, NY (1995).
9. H. Ludwig, J. Hogan, R. Jaluka, D. Lowenstern, S. Kumaran, A. Gilbert, A. Roy, T. R. Nellutla, and M. Surendra, "Catalog-Based Service Request Management," *IBM Systems Journal* **46**, No. 3, 531–548 (2007, this issue).
10. C. Y. Baldwin and K. B. Clark, *Design Rules, Volume 1: The Power of Modularity*, The MIT Press, Cambridge, MA (2000).
11. J. Koehler, G. Tirenni, and S. Kumaran, "From Business Process Model to Consistent Implementation: A Case for Formal Verification Methods," *Proceedings of the Sixth International Enterprise Distributed Object Computing Conference* (2002), pp. 96–106.
12. WebSphere Business Integration Server Foundation, IBM Corporation, http://www.ibm.com/software/integration/wbisf/features/.
13. B. Portier and F. Budinsky, *Introduction to Service Data Objects*, IBM Corporation (2004), http://www.ibm.com/developerworks/java/library/j-sdo/.
14. L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE* **77**, No. 2, 257–286 (February 1989).
15. DB2 Alphablox, IBM Corporation, http://www.ibm.com/db2/alphablox.
16. Unified Modeling Language, Object Management Group (2007), http://www.uml.org/.
17. Java 2 Platform, Enterprise Edition, Version 1.4 Documentation, Sun Microsystems (2005), http://java.sun.com/j2ee/1.4/docs/.
18. *Web Services Description Language 1.1 (WSDL 1.1)*, Worldwide Web Consortium (2001), http://www.w3.org/TR/wsdl.
19. *Business Process Execution Language for Web Services Version 1.1* (2002), http://www.ibm.com/developerworks/library/specification/ws-bpel/.
20. F. Leymann and D. Roller, *Production Workflow: Concepts and Techniques*, Prentice Hall PTR, Upper Saddle River, New Jersey (2000).
21. *Workflow Management Coalition: Terminology & Glossary*, Workflow Management Coalition (1999), http://www.wfmc.org/standards/docs/TC-1011_term_glossary_v3.pdf .
22. J. A. Zachman, "A Framework for Information Systems Architecture," *IBM Systems Journal* **26**, No. 3, 276–292 (1987).
23. OMG Model Driven Architecture, Object Management Group (2003) http://www.omg.org/mda.
24. K. Gottschalk, S. Graham, H. Kreger, and J. Snell, "Introduction to Web Services Architecture," *IBM Systems Journal* **41**, No. 2, 170–177 (2002).
25. Q. Wu, C. Pu, A. Sahai, R. Barga, and G. Jung, "DSCWeaver: Synchronization-Constraint Aspect Extension to Procedural Process Specification Languages," *Proceedings of the International Conference on Web Services (ICWS '06)* (2006), pp. 320–330.
26. J. Sauvé, A. Moura, M. Sampaio, J. Jornada, and E. Radziuk, "An Introductory Overview and Survey of Business-Driven IT Management," *Proceedings of the First IEEE/IFIP International Workshop on Business-Driven IT Management (BDIM '06)* (2006), pp. 1–10.
27. V. Tosic, "The 5 C Challenges of Business-Driven IT Management and the 5 A Approaches to Addressing Them," *Proceedings of the First IEEE/IFIP International*

Workshop on Business-Driven IT Management (BDIM '06), (2006) pp. 11–18.

28. M. Brenner, "Classifying ITIL Processes; A Taxonomy under Tool Support Aspects," *Proceedings of the First IEEE/IFIP International Workshop on Business-Driven IT Management (BDIM '06)* (2006), pp. 19–28.
29. V. A. Danciu, "Formalisms for IT Management Process Representation," *Proceedings of the First IEEE/IFIP International Workshop on Business-Driven IT Management (BDIM '06)* (2006), pp. 45–54.
30. C. Mayerl, F. Troscher, and S. Abeck, "Process-Oriented Integration of Applications for a Service-Oriented IT Management; Integrated IT Management Architecture," *Proceedings of the First IEEE/IFIP International Workshop on Business-Driven IT Management (BDIM '06)* (2006), pp. 29–36.

Accepted for publication March 14, 2007.
Published online July 13, 2007.

Santhosh Kumaran
IBM Research Division, Thomas J. Watson Research Center, 1101 Kitchawan Road, Yorktown Heights, NY 10598 (sbk@us.ibm.com). Dr. Kumaran leads a team of researchers in the area of model-driven business integration. His research interest is in using formal models to explicitly define the structure and behavior of an enterprise and employing these models to integrate, monitor, analyze, and improve its performance.

Pete Bishop
IBM Integrated Technology Division, 11502 Burnet Road, Austin, TX 78758 (ellisb@us.ibm.com). Mr. Bishop is a senior process architect in the Information Services department of the Technology and Integration Management competency of the Integrated Technology Delivery organization. He has been involved in strategic process work for over 13 years, primarily in the areas of IT systems management and IT service delivery management.

Tian Chao
IBM Research Division, Thomas J. Watson Research Center, 1101 Kitchawan Road, Yorktown Heights, NY 10598 (tian@us.ibm.com). Ms. Chao is a senior software engineer in the Business Informatics department at the Watson Research Center. She has a B.A. degree from the National Taiwan University and an M.S. degree in computer science from Virginia Polytechnic Institute. Her research interests include service-oriented computing technologies in business performance monitoring and management using the model-driven approach and business-process security. Ms. Chao holds several patents. She has received both invention and project-related awards and authored many journal and conference papers.

Pankaj Dhoolia
IBM Research Division, India Research Lab, 4 Block-C Institutional Area, Vasant Kunj, New Delhi, India 110070 (pdhoolia@in.ibm.com). Mr. Dhoolia works in the area of business informatics at the IBM India Research Laboratory. His research interest is in using formal models to describe and implement the deep structure of adaptive businesses.

Prashant Jain
IBM Research Division, India Research Lab, 4 Block-C, Institutional Area, Vasant Kunj, New Delhi, India 110070 (prashantjain@in.ibm.com). Mr. Jain is the manager of the Business Development and Solutions department at the India Research Lab. He has M.S. and B.S. degrees in computer science from Washington University, and a B.A. in physics from The College of Wooster. He has over 11 years experience in software research, project management, business consulting, and design and development of software products and services. Mr. Jain's primary areas of interest are design patterns and model-driven business-process integration, specifically focusing on tools and technologies facilitating model-driven business transformation. He is an author of a book on design patterns.

Rajesh Jaluka
IBM Integrated Technology Division, 2455 South Road, Poughkeepsie, NY 12601 (rjaluka@us.ibm.com). Mr. Jaluka is a senior IT Architect at the IBM Technology and Integration Management competency. As the lead architect of the Service Delivery Management project he is working on the methodology for streamlining the delivery of IT services, building content and workflows for service catalogs, and deploying SDM tools. He has 18 years of experience in developing and deploying IT solutions for the manufacturing, finance, telecommunication, and services industries.

Heiko Ludwig
IBM Research Division, Thomas J. Watson Research Center, 19 Skyline Drive, Hawthorne, NY 10532 (hludwig@us.ibm.com). Dr. Ludwig is a research staff member at the Watson Research Center. As a member of the Service Delivery Management department, he is currently working on service componentization, service-delivery-management platforms (which include aspects of large-scale, loosely coupled distributed systems), federated workflow management, and management of the variability of processes and configurations. He is also involved in SLA and policy management. Earlier, when he was at the IBM Zurich Research Laboratory he worked on cross-organizational process management. He has a Master's (Diplom) degree and a Ph.D. degree in information systems (Wirtschaftsinformatik) from Otto-Friedrich University in Bamberg, Germany. More information on Dr. Ludwig can be found at http://www.research.ibm.com/people/h/hludwig/.

Ann Moyer
IBM Global Technology and Integration Management, 24 Niles Drive, Woodstock, NY 12498 (annmoyer@us.ibm.com). Ms. Moyer is a Distinguished Engineer with over 26 years in the IT service industry. Her expertise is in e-business and on demand technical solution development, systems management, system integration, IT service catalogs, call center technology, and system support.

Anil Nigam
IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (anigam@us.ibm.com). Dr. Nigam, a research staff member in the Business Informatics department, joined the IBM Research Division in 1981, soon after earning a Ph.D. degree in computer science at the University of Rochester. His research at IBM spans a wide range: VLSI design systems, parallel-processing architectures and database machines, logic programming and databases, knowledge representation, qualitative reasoning, operational business modeling, and business design. He has received many awards, including Research Division Awards, a Research Commercialization Award, an IBM Consulting Group Engagement Excellence Award, and a Technical Group Award. Dr. Nigam has also published extensively and holds a number of patents. ■