Catalog-based service request management

H. Ludwig
J. Hogan
R. Jaluka
D. Loewenstern
S. Kumaran
A. Gilbert
A. Roy
T. Nellutla
M. Surendra

To manage the delivery of services competitively on a large, global scale, an IT (information technology) service provider must efficiently use service delivery resources—in particular, skilled service delivery teams. Service requests form a large and important component of the management of a client's IT infrastructure. Currently, the fulfillment of IT service requests is often managed on a per-account basis. Service-delivery teams fulfill service requests according to account-specific processes by using an account-specific service-request-management environment, making it difficult to leverage the skills of the various delivery teams for multiple accounts. The service delivery management platform (SDMP) uses reusable service components that can be performed by multiple delivery teams and can be assembled into service compositions to which multiple clients can subscribe. The SDMP catalog is the information repository that manages service components, composition, providers, and subscriptions and is used by the service-request runtime environment to implement specific customer service requests. In this paper, we describe a catalog-based architecture for service delivery management and demonstrate how its use can provide a global service-delivery organization with a platform for achieving significant productivity gains.

INTRODUCTION

The business of providing IT (information technology) services is competitive and global and has a wide range of market participants, from traditional IT service providers to telecommunication companies expanding their portfolios to recent entrants from emerging economies, such as India and China. To remain competitive in the delivery of IT services on a global scale and manage multiple accounts, an IT service provider must efficiently use service delivery resources such as IT infrastructure, best practices, and, in particular, skilled service delivery teams in order to achieve economies of scale.

Service requests, typically issued by the employees of a client organization, constitute a large and important aspect of the management of a client's IT

infrastructure, typically representing half or more of the volume of management work. As a result, their timely and efficient handling has a great deal of value both to the service provider and the client. These services, such as provisioning a new server or installing a patch or new application, typically require both automated and manual fulfillment. This paper focuses on the efficient handling of this work.

Currently, service requests are most often managed in an isolated manner for each customer account. Service-delivery teams fulfill service requests according to account-specific processes using an account-specific service-request-management environment. This "silo" approach is necessitated by differences in the delivery processes and procedures imposed by various customers. Until now, no technology platform has been able to organize and optimize the IT delivery teams or their work for multiple silos. This limitation has been difficult to overcome, not only due to customer variability but also because the work of delivering IT is structured into large, monolithic processes, making it difficult to change one part of the work without re-engineering the entire process, as well as processes preceding and following it.

This limitation makes it nearly impossible to leverage the skills of delivery teams or implement best practices beyond individual accounts. Conversely, any attempt at simple standardization of processes for all accounts and service delivery teams does not address customer-specific requirements that are important for their business. Furthermore, service delivery teams in large service organizations such as the IBM Integrated Technology Delivery (ITD) organization have different ways of organizing themselves efficiently to perform their work and different technologies to manage the internal elements of their work. A service-delivery-management platform for a large IT service provider must be able to reap the benefits of standardization of delivery processes without sacrificing either the specific requirements of the customer or the specific efficiencies of individual service delivery teams.

SDMP addresses this challenge by providing reusable elements of services, called *atomic services*, as components. For example, one such service is the installation of an operating system (OS) image. The atomic services have standard inputs and outputs (e.g., the OS version) that allow them to be assembled into service compositions that are meaningful in the context of a client service request. Service compositions can be shared among multiple clients. For example, a composition for the provisioning of a new database server may contain the OS-installation atomic service as a component. Finally, service compositions can be customized for each account without compromising the overall reusability of the atomic services. Atomic services may be provided by internal or external service providers, and multiple service providers can be capable of rendering the same atomic service. Many service providers utilize an IT service management (ITSM) platform to deliver their services. SDMP can be integrated as a core Process Manager component of the Tivoli ITSM platform.

The SDMP catalog is the core information repository that manages the types of atomic services (and specifies which delivery teams or third-party vendors can perform them), the service compositions, the account subscriptions, and specific customizations. The SDMP catalog also provides the basis for generating the content of client-facing customer service catalogs, which are used by employees of client organizations to request a specific service. The client-facing customer-service-catalog system can submit service orders to the SDMP or to other service delivery systems for simpler services, for example, building maintenance or catering.

The content of the SDMP catalog can be managed by using a set of visual tools in the context of standard ITSM practices. Service-delivery catalog entries can be imported from account service catalogs maintained by delivery teams. The service-request runtime environment accesses the repository to implement and manage specific customer service requests.

The SDMP catalog-based architecture for service delivery management provides a global service delivery organization with a platform to achieve significant productivity gains while still being able to accommodate client-specific requirements. Catalog-driven service delivery management is a new approach to managing the delivery of IT services in today's geographically dispersed, outsourced, and highly matrixed environments. At the heart of the approach is a catalog of service definitions tailored to the client enterprise's requirements but based on

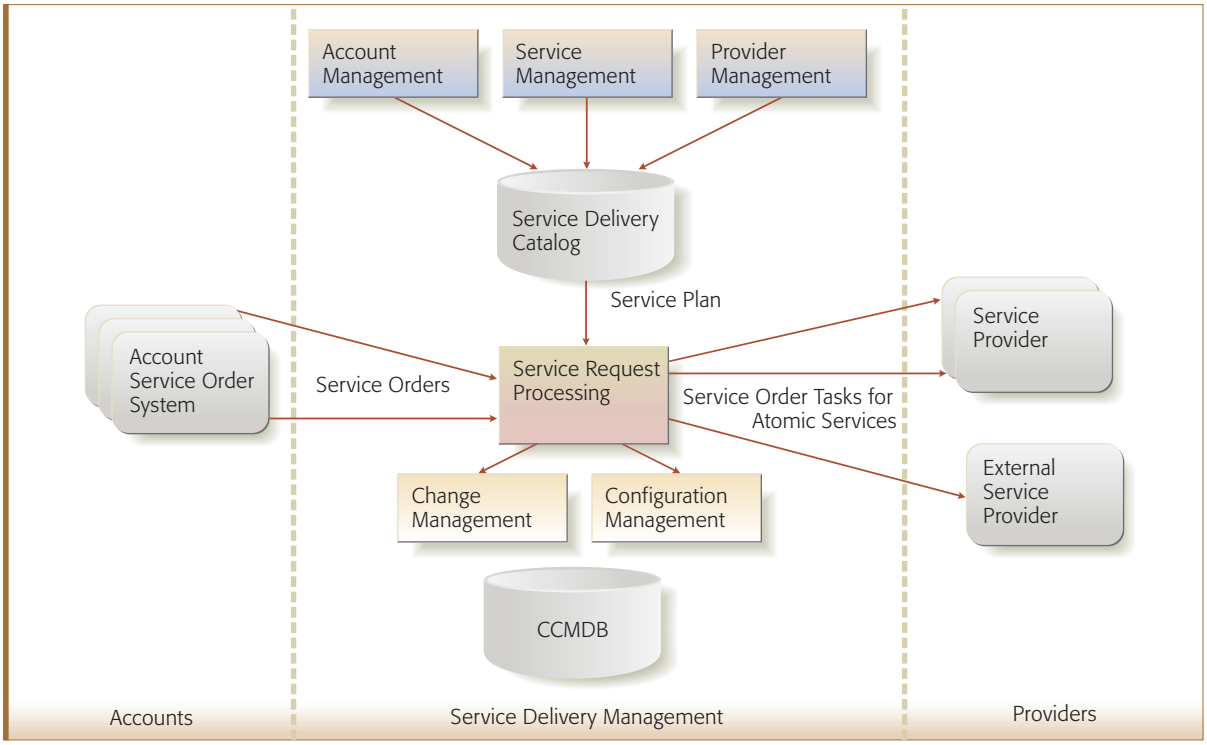

Figure 1
Service delivery management architecture

a standard set of delivery best practices. This allows for client customization to meet local requirements while ensuring alignment with proven efficient delivery mechanisms.

This paper is organized as follows. The next section outlines the main issues and requirements of service request processing. The model of creating service components and service composition is then introduced. In the following three sections, we elaborate on the organization of the service-catalog data structures and the life cycles of the catalog components. This is followed by a discussion of the use of the catalog to align business goals, the process and challenges of catalog implementation, and related work. The paper concludes with a discussion of the results we have achieved and future work.

MANAGING SERVICE REQUESTS IN LARGE-SCALE IT SERVICE DELIVERY ENVIRONMENTS

In this section, we introduce a service-request-management architecture and present some of the requirements of a large-scale service request infrastructure, as well as some of the challenges involved in constructing it.

Service-request-management architecture

IT service providers manage service requests in a service-delivery-management environment that comprises a number of components, as outlined in *Figure 1*.

At the center of the delivery management environment is the *service request processing* component. This component receives service orders from clients, processes them administratively (e.g., checks the entitlement of the request, prices the request, connects to billing, etc.), and devises a service plan that commissions service order tasks to service-providing teams or external service providers who provide the service. Details of the SDMP service-request-processing component are discussed in Reference 1. The *service plan* is a workflow specification that defines the tasks required to implement the service request and the control and data flow related to these tasks. Based on the service plan, service providers can be chosen for each task.

ITSM components are used to operationally manage services which have an effect on the IT infrastructure that is provided to the client. Service requests typically pertain to change management, as parts of the IT infrastructure are typically changed in the context of fulfilling the service request. Configuration management maintains the configuration of the IT service infrastructure. Both components are based on the IBM Tivoli Change and Configuration Management Database (CCMDB), which records the status and development of the IT infrastructure.

The *service delivery catalog* is the metadata repository that makes service plans, client subscriptions, and service providers and their capabilities available to the service request processing component. Whereas the service request processing component provides the operational management of a service request (deciding on specific service providers, scheduling the service plan, etc.), the service delivery catalog is the tactical and, in parts, strategic service-delivery-management instrument. The service delivery catalog is managed by three components, which address the different aspects of service request management: service management, service provider management, and account management.

Service management deals with defining atomic service definitions and service compositions. This function is instrumental in analyzing and defining the scope of the services that are being produced by the service organization and how they are combined into services that provide value to customers. *Service provider management* handles resource and capacity management, addressing the following issues: Which atomic services are being provided by internal delivery teams and which are being contracted out to external service providers? Which teams are devoted to each type of service and how much service capacity is needed for specific services? In addition, specific customer requirements relating to location of service providers, time zone availability, security clearances, and so forth have to be taken into account. *Account management* defines which services are available to each client. In addition, it captures the specific requirements of an account for the manner in which particular services are to be delivered, for example, specific preferences related to the location of service providers for particular tasks. Also, the specific service level requirements relating to key performance indicators (KPIs), for example, service turnaround times, are captured for each account.

An IT service provider organization can use KPIs not only for evaluating customer-service-level objectives but in general to stay focused on the service delivery goals that would differentiate them from their competitors. The following are some examples of service delivery KPIs: reduction of turnaround time, enhancement of delivery team throughput, reduction of SLA (service-level agreement) breaches, and improvement of QoS (Quality of Service). It is not sufficient to produce historic reports to measure actual performance against a benchmark. In order to achieve or surpass the benchmarks, it is necessary to manage the performance of delivery teams actively at a more granular level (i.e., the level of atomic services or even tasks) and to take corrective actions at every step.

In order to manage service delivery teams at the atomic service level, it is necessary to define internal benchmarks at that level, apart from the benchmarks that represent the fulfillment of the entire service that are primarily from requester's point of view. For example, apart from the turnaround time to fulfill a request to build a new database server, the delivery teams need to define turnaround times for building a server, allocating storage, installing and configuring database software, connecting a server to the network, and testing a server. Measuring at this level helps delivery teams identify bottleneck areas, marketing opportunities, automation opportunities, and so forth.

Challenges and requirements of a large-scale service request infrastructure

IT service providers such as IBM ITD have to deal with numerous challenges that are significantly different from those faced by an in-house, enterprise IT organization. Some of these challenges are described in the following:

- *Limitations of enterprise software*—There are hardly any software products in the marketplace for managing the delivery of IT services that have been designed to accommodate outsourcing. One of the most commonly ignored design principles is multitenancy: the systems are designed to manage the services for a single company. There is no notion of separating the service provider's organizational structure from that of those who are

being served. This often requires IT service providers to build separate instances of IT systems, one for each customer, to manage the delivery of IT services. This not only increases the infrastructure cost, but also creates difficulties for the people fulfilling the services. They have to log into multiple instances of such systems to determine the work they have to perform. Furthermore, over time, these systems end up using different releases of the software. Another important design shortfall involves the inability to handle multiple customers' business calendars to calculate SLA.

- *Customer-specific processes*—When IT services are outsourced, most service providers inherit legacy processes from their customers. There is very little transition time or budget to streamline or standardize these processes. As a result, the IT service provider ends up dealing with a plethora of processes. One of the biggest challenges is to find a tool that can promote reusability of the processes and at the same time handle customer-specific variations.
- *Using customer-specific systems*—Along with processes, service providers also inherit legacy tools. In many cases, the customer's employees are involved in submitting requests, approving changes, and performing work. In order to minimize disruption, customers mandate (in outsourcing contracts) the use of existing tools. This also limits the service provider's ability to standardize on ITSM tools. Further, the time and cost involved in integrating the customer tools are so high that the service providers ultimately use the customer-mandated tools to serve the contract.
- *Account-dedicated fulfillment teams*—The issues just mentioned not only increase the service provider's cost to train its employees in multiple processes and tools but also limit its ability to share resources among accounts, reducing the flexibility of delivery teams. These characteristics inhibit the sharing of resources and best practices that could be used to achieve economies of scale.
- *Difficulties in benchmarking across accounts*—Given the differences in tooling and the specific processes for each account, gathering statistical information on delivery effectiveness across all accounts is nearly impossible. Different systems provide different instrumentation for gathering base statistics, and processes that provide similar business value for customers (such as patch installation or server provisioning) may have different sets of steps, which makes gathering standardized performance data difficult. As a result, the performance of different teams can be difficult to compare.

In light of these challenges for service providers, we have derived a set of requirements that a service-request-management system must address in order to overcome some of the problems mentioned. This applies in particular to the service delivery catalog, as it is the tactical management instrument for service requests.

The direct dependency between service delivery teams and accounts must be removed. For that purpose, we need a model of service processes that allows us to define the contributions that service teams can make in an account-independent way as atomic services, as discussed in the Introduction. Service process definitions must be sharable between accounts.

The account-specific requirements of services must be addressed with a minimal impact on service teams. While ideally the same processes would be used to service all clients, there are often legitimate customer requirements for modifications of best practices processes, for example, to connect to the customer's internal systems or to notify another service provider about the progress of the service process. The service delivery catalog must be able to capture these specific requirements, and the runtime system must be able to execute the changed process with minimal impact to existing service team processes. Different customers must be able to receive service at different service levels.

Service providers, internal service delivery teams as well as external service providers, must be able to use their own internal processes to perform their part of a service process, independent of what other service providers in the process use. This requires the precise definition of interfaces to service delivery teams. Based on this isolation, service providers can make improvements to their internal processes without affecting the overall service. Furthermore, all service provider teams may not be on the same level of software and tooling, which requires a complex rollout in a large, global delivery organization. Finally, as skill levels and labor costs may be different in different data centers, various degrees of automation and different tooling may be the best

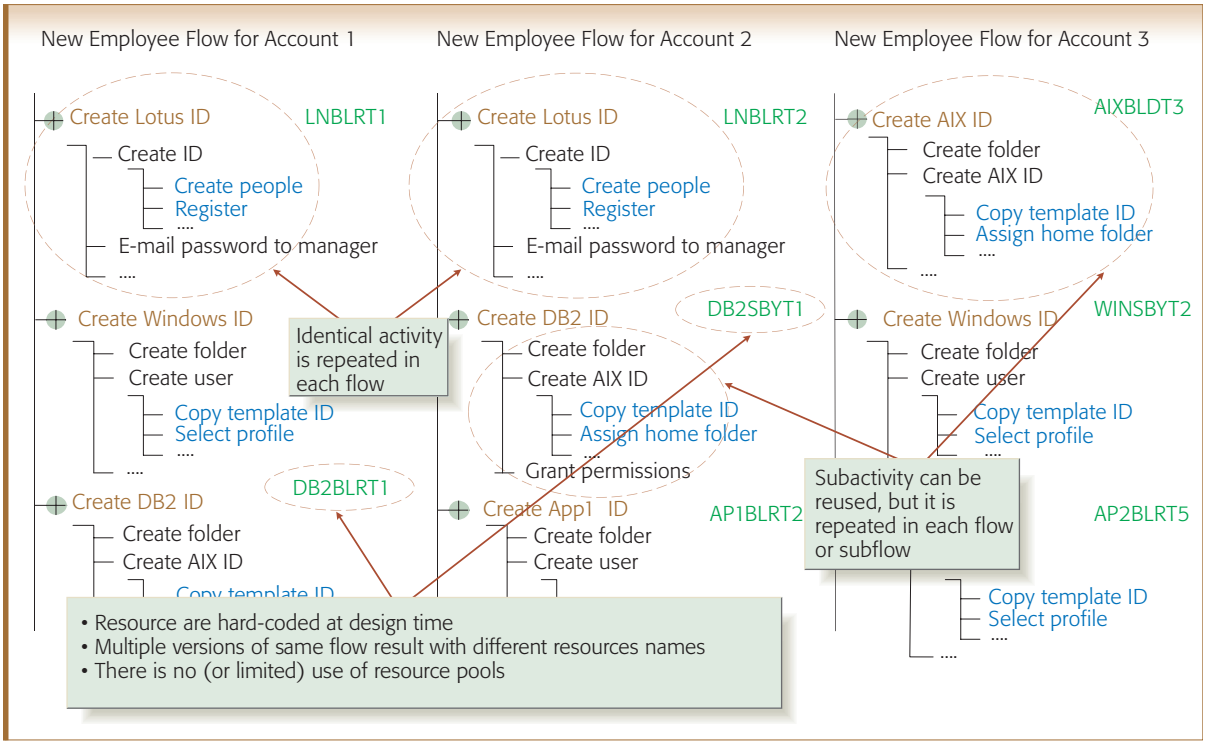

Figure 2
Service workflows containing overlapping elements

support for the delivery staff. All contents of the service delivery catalog must be subjected to proper governance processes, as these are central to the service product strategy of the delivery organization, the capacity management of delivery teams, and the contractual obligations toward clients.

In the context of this paper, we address these requirements from the point of view of the service delivery catalog.

SERVICE COMPOSITION MODEL FOR LARGE-SCALE SERVICE DELIVERY

In the previous sections, we discussed the need to create service delivery components to enable delivery teams to work beyond the process structure of specific accounts. In this section, we exemplify this issue with scenarios for creating user IDs for a new employee, as shown in ***Figure 2***. The types of IDs to be created and the necessary access approvals can greatly vary from one customer to another. When the process and steps are captured in a workflow system, the result is variability in the new employee workflows. Instead of defining all the steps in a single workflow, one could break down the flow into multiple flows ("microflows"), one for each type of user ID, for example DB2*, AIX*, or NT. We use the term *atomic services* to describe services in which the breakdown is performed in such a manner that the service delivered through a microflow can be owned by a single logical team and can also function as a "black box." For a microflow to qualify as an atomic service, it must have well-defined inputs needed to perform the service, well-defined deliverables, and well-defined responses. A set of different atomic services for different types of user IDs can serve as building blocks for service delivery teams to compose additional new hire workflows for new customers. *Composition* is simply the specification of which atomic services help fulfill a complete service and in what order. We use the term *composite services* for such compositions of atomic services. Similar to an atomic service, a composite service must have a well-defined input message needed to perform the service, well-defined deliverables, and a well-defined response message.

This methodology can be applied to all services performed by delivery teams in order to make the

delivery processes reusable. For example, the allocate-space service can be reused for creating a file system, creating a database, or installing an application. If one could build a catalog of such reusable atomic services, creating new or customizing existing IT services would become very easy. A plan could be used to compose higher-level services, specifying which atomic services were to be used in completing an activity and their order. The plan could be standardized, customer-specific, contract-specific or even industry-specific. For example, a standard build-DB2-subsystem composition could make use of the following atomic services: create file system, push DB2 image, create DB2 instance, and create DB2 database. A customized build-DB2-subsystem composition would make use of an additional atomic service, allocate SAN (storage area network) storage, if SAN-attached storage were a requirement.

Service providers often have to deal with teams that are dedicated for some customers and shared for others, and some portions of the work may be performed by customer teams while others are outsourced to partners or third-party providers. The workflow tools cannot address such variations, and it is often necessary to create multiple versions of the same workflow to accommodate these variations. While decomposition promotes reusability, decomposition without decoupling resource assignment from the actual workflow can severely limit reuse of processes among customers.

A delivery catalog of atomic services and composite services along with the ability to dynamically assign atomic services to different performer teams or service providers for each order or request is critical for an efficient service delivery organization.

There are several other benefits of this approach. The use of components allows KPIs to be defined at multiple levels, for example, the composite-service and the atomic-service levels. Using components also enables measurement of the performance of service delivery teams at a more granular level. Decoupling providers from workflow increases the resiliency of service delivery: if one team has a higher workload or is unable to perform due to an outage, any other team can pick up the work and deliver the same results. A better use of teams can also result: a scheduling engine can optimize workload among multiple service providers as opposed to limiting itself to the resources of a single service provider.

The service delivery catalog facilitates the service composition model by providing the repository for storing the artifacts and the service management components to create them and manage their life cycle. In addition, it provides an account perspective and a service provider perspective.

MANAGING SERVICES

At the core of the service delivery catalog is the management of services, both atomic and composite.

Representing atomic and composite services

An *atomic service definition* (ASD) specifies a type of atomic service in the service delivery catalog and has the following set of attributes: a service identifier, a service name, and a version number, representing the current version of the service with this service name. Different versions can be active in parallel. Hence, each new version of the ASD has its own service identifier. ASD attributes also include one or more service categories, keywords (which help in searching for a particular ASD), and a *deliverable* attribute (containing a natural language description of what that service does and what effect its execution has).

The *qualifying characteristics* attribute describes requirements for service providers who want to implement the service. They are phrased in plain natural language and are meant to be read by service providers before signing up to provide the service defined by a particular ASD and to be checked by the delivery organization before certifying a service provider for the ASD. *Formal characteristics definitions* are machine-readable specifications of properties of an ASD, such as location and business hours. These are akin to variable definitions. Each service provider is expected to provide values for these characteristics. *Constraints* can be defined for these formal characteristics definitions. They are expressions in a formal constraint language and are evaluated automatically when service providers sign up for ASDs. Formal characteristics definitions together with constraints provide an automated approach for managing qualifying characteristics. The *planned time* attribute defines the typical time in which an atomic service defined by this ASD will be delivered. The real time or scheduled time

depends on the specific service provider and the specific instance of the service.

The *abstract interface definition* attribute defines the way atomic services are to be invoked. Following the service-oriented approach, the SDMP relies on the Web Services Description Language (WSDL) for the specification of an interface. However, an atomic service in the context of service delivery is not directly equivalent to the concept of a service in a service-oriented architecture (SOA). Each atomic service corresponds to one operation within a WSDL port type (defined as a named set of abstract operations). Hence, the abstract interface definition comprises a WSDL file, the name of a port type, and the name of an operation. Furthermore, the WSDL file does not specify binding-level information, as this type of information is relevant for the binding to a particular service provider. This separation of port-type-level WSDL and binding-level WSDL, which often includes the port-type-level WSDL through references, is a common approach when designing service-oriented systems.

ASDs are composed into service compositions making them independent of specific service providers. Service compositions, like ASDs, have a service identifier, service name, service category, service description, and keywords. The planned time of a service composition is an expected turnaround time based on the planned time estimates of the ASDs on which the service composition is based. Each service composition has a version number. Different versions of a service composition with the same name receive different service identifiers. Service compositions refer to a specific version of an ASD. Hence, if a new version of an ASD is being included in a service composition, this also entails a new version of the service composition itself.

The core of the service composition is the *service plan*. This is the description of the workflow to be executed to implement the service. The composition is defined in a WS-BPEL[2] (Web Services Business Process Execution Language) specification, using ASDs that are available in the service delivery catalog. The service plan refers to the operations defined in the abstract service specifications of the ASDs. The service plan is retrieved by the SDMP runtime system at the time of a service request and can be modified by the service coordinator before the service request instance is executed.

The inception date of a service composition is the time at which a particular version of the composition can be made available to accounts; the termination date marks the time after which no service requests may be issued for this service composition version. The *cancelable* attribute defines whether the execution of the service may be canceled by the client while the service process is in progress. For some types of services this is necessary.

The *entitlement policy* of a service defines the circumstances under which the service may be requested. The *abstract interface* defines the parameters of the service request that must be supplied by the client when the service is requested. It corresponds to its ASD equivalent and is also based on port-type-level WSDL specifications.

Service compositions can have a service-level definition associated with them to define the QoS of the composition. It comprises a set of QoS attribute definitions, the KPIs of the service, and the definition of service-level goals as constraints on the QoS attributes. A service level is also associated with a cost of delivery to a client, which may vary with different service levels. A service composition may be offered and executed at different service levels.

Life cycles of atomic and composite services

ASDs and service compositions are the core concepts of the service delivery catalog. To a large extent, their design shapes the way in which a service delivery organization works. Due to this importance, creation and change of ASDs and service compositions are subjected to a governance process to ensure that all relevant stakeholders in the service delivery organizations (account executives, marketing and general management, as well as delivery teams and liaisons to external service providers) can provide input when changes occur.

Figure 3 outlines the governance process of an individual ASD. The representation is in the visual format of the WebSphere* Business Integration Modeler. There are two roles involved: the *service designer* is the team or individual in charge of creating and modifying the content of the ASD as outlined in the previous sections. All creation and editing tasks are assigned to it. The service designer also decides when modifications become active. The *certification board* evaluates ASDs when they are

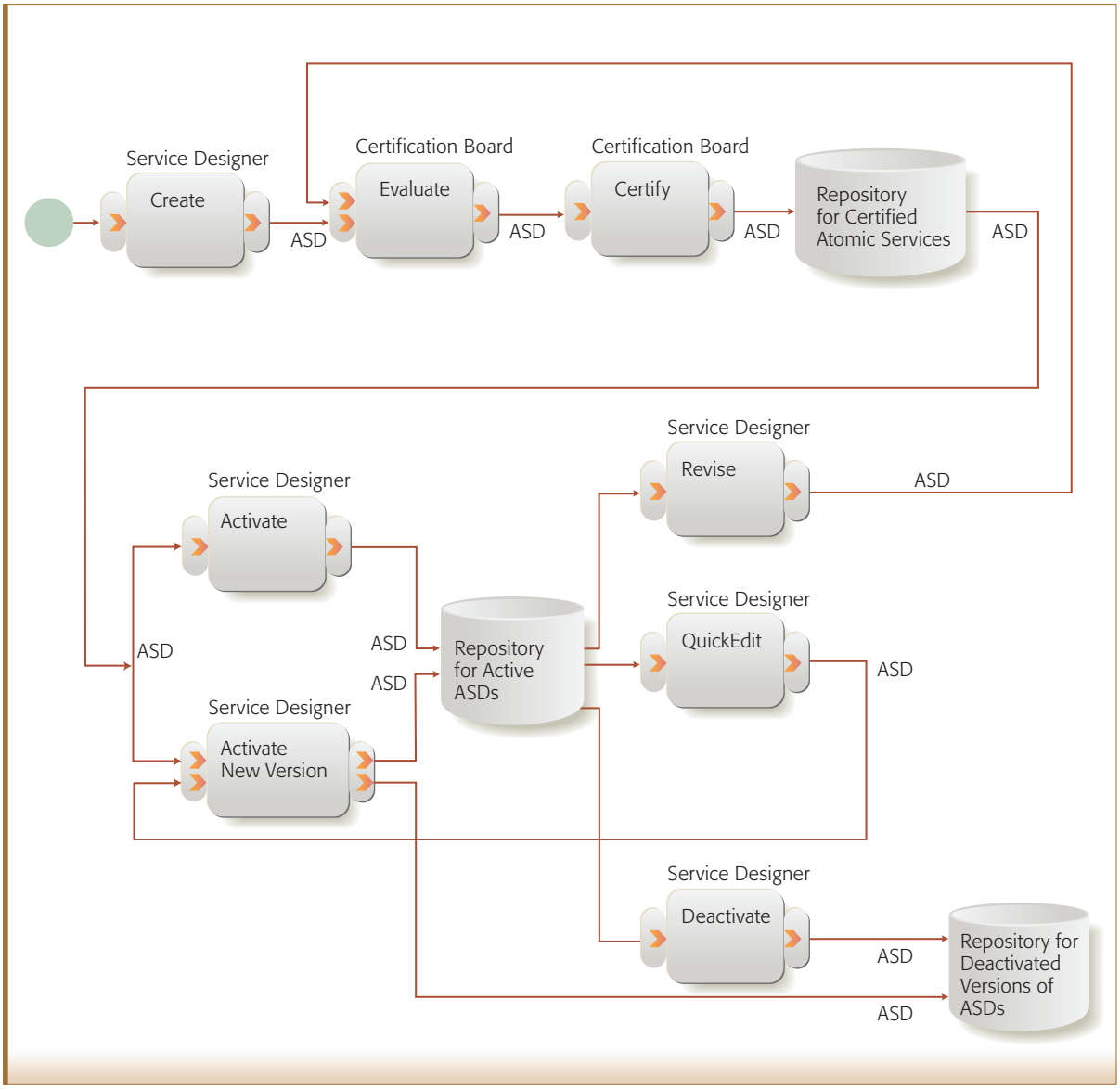

Figure 3
ASD governance process

created and changed. It represents all of the stakeholders in this particular ASD.

In the first step, a new ASD is created by the service designer. The certification board then evaluates the impact of the proposed ASD. This may include determining which service provider is able to implement this ASD and in which service compositions it may be used. If approval is given (or the ASD is sent back to the service designer), it is available to be activated by the service designer based on the requirements of the service compositions and the availability of service providers implementing the ASD. Activation puts the ASD into potential use in a service composition.

There are two modes of editing. Minor changes (e.g., the change of a description) can be applied in a "quick edit" step, requiring no further approval. Changes with impact on service providers and service compositions are performed as revisions. To put a revision into service, a new evaluation and certification must be performed by the certification board. In particular, the impact of the atomic service

on service providers must be assessed. When a revision is activated, the currently active revision becomes deactivated. This means that it still can be used for performing existing service compositions but cannot be included in new ones. Hence, multiple versions of the same ASD may exist in parallel.

Each service composition has a maintenance flow that corresponds exactly to the one of ASDs depicted in Figure 3. Also, the repository of activated ASDs provides input to the creation step. A certification board evaluates and certifies service compositions. The objective of the evaluation is to ensure that revised service compositions meet customer needs and that implementations are available for the atomic services used in the composition. Deactivation of an ASD in a service composition while the composition is active does not deactivate the composition itself. Deactivation of the service composition version prevents further submission of service requests using that composition but does not have an impact on service requests currently being processed.

MANAGING SERVICE PROVIDERS: REPRESENTING SERVICE PROVIDERS AND THEIR CAPABILITIES

As mentioned previously, we use the term "service providers" in our model for teams internal to the service delivery organization as well as teams of external organizations providing service. Although different information must be captured for external service providers to facilitate accounting, billing, and so forth, we do not take these financial aspects into account for our service-request-management system.

We use a simple content model for service providers. A service provider has a unique provider identifier. Service providers internal to the IBM ITD organization have a Global Services Delivery Center (GSDC) ID, which encodes the specific organization and provides a key to further information about the organization through LDAP (Lightweight Directory Access Protocol)[3] and other internal systems. External service providers have a vendor ID, which provides a key to purchasing systems and other ERP (enterprise resource planning) and accounting systems. Each service provider has a name, description, and location attributes. This is important for on-site services to clients. A number of contacts can be defined; namely, employees of this service provider who are available for discussions.

Each service provider can be associated with a set of *service provider capabilities* (SPCs). SPCs are associations between an ASD and a specific service provider outlining in specifics how the service provider will fulfill the service. An SPC instance relates one service provider to one ASD. This relationship is captured with data containing the service identifier of an ASD and a service provider identifier. It has an inception date and a termination date, defining the beginning and end of the period in which the service provider accepts requests related to this atomic service. The *cancelable* attribute defines whether the atomic service as provided by this provider can be canceled after being started.

The *interface* attribute defines in specifics how to invoke the atomic service implementation of the provider. The interface contains a binding-level WSDL specification that refers to the abstract WSDL of the ASD to which this SPC relates. Because a WSDL specification may contain multiple services and bindings for different protocols, the names of the services and bindings are also defined. In conjunction with the WSDL specification of the ASD and its port type and operation definition, the interface definition of the SPC is sufficient to allow the service delivery system to bind to the specific atomic service implementation of the service provider.

In the *qualifying characteristics* attribute, a service provider provides the information that is specified by the ASD, either as a natural language entry, if the requirement is phrased in natural language, or as values provided for the formal characteristics definitions of the ASD. Based on this information, the service delivery organization can verify whether the service provider meets the requirements of the ASD. A service provider can have any number of SPCs, and each ASD may be associated with multiple service providers through multiple SPCs.

Signing on and integrating service providers

Frequently, it may be beneficial for a service provider to retain internal service delivery teams for typical capacity but to rely on external service providers to satisfy peaks in service requests. In a large distributed service organization such as ITD, there are multiple delivery teams for common atomic services in various delivery centers in different geographic areas; whereas, other delivery

teams may be unique in their capabilities, specializing in a particular, narrow technology domain.

Given the large number of service providers, internal and external, we assume that the core information on service providers in the service delivery catalog will be maintained by the providers themselves, without any further approval process. However, external service providers might need to undergo additional scrutiny outside the scope of the service delivery catalog to be approved as preferred vendors. Whereas being listed as a service provider is simple, becoming a service provider for a particular ASD requires scrutiny by the catalog administrator. SPCs are subject to a detailed governance process, as outlined in ***Figure 4***.

In the registration step, a service provider creates an SPC object referring to the ASD in question and the service provider's own item in the service delivery catalog. In this step, information on the binding-level WSDL specification and on qualifying characteristics must be specified by the service provider. The catalog administrator then evaluates the SPC, verifying that the binding-level WSDL is valid and that the qualifying characteristics of the provider match the requirements defined in the ASD. If certification is granted, the SPC can be activated by the certification board, not by the service provider itself. If activated, a request for atomic services can be routed to this service provider by the service-request runtime component, using the binding information defined in the WSDL file of the SPC.

Changes to SPCs are managed in a manner similar to ASD and service-composition changes. A simple editing step can be used for minor changes. A formal revision process with evaluation and recertification must be used for larger changes that impact the operation of the atomic service. Changes in an ASD require a new SPC, defining how to deal with this new version of the service. Atomic services can be provided in multiple versions by the same service provider in parallel, each requiring a separate SPC and, potentially, a modified implementation at a different endpoint. At runtime, the service delivery manager chooses one of the service providers, having registered an SPC at his own discretion for the execution of each of the atomic services within the composition of the offered service.

MANAGING ACCOUNTS: REPRESENTING OFFERED SERVICES

The set of services to which an account has subscribed is referred to as its *offered services*. Each offered service associates a service composition with an account. An offered service has a service composition ID, an account ID, a client ID, an inception and termination date, and an alerting threshold.

An alerting threshold defines the percentage level of SLA compliance at which the client wants to be notified about potential of infraction. For example, if the maximum turnaround time for provisioning a server is 5 days, the client may want to be notified if the average exceeds 4 days; in this case, the alerting threshold is set to 80 percent.

Each account can define default service providers for ASDs of the service compositions in its offered services. Accounts might have location preferences or restrictions defined by government regulations. Providers might require security clearance for on-premise services, which a particular external service provider or members of an internal service delivery team may already have.

Using these offered services, account managers can tailor the use of service compositions to the needs of their accounts. In addition, there is always the opportunity to tailor a new service composition to the requirements of a specific account, based on the set of ASDs available in the service delivery catalog. By reusing the ASDs, services can be customized without causing additional complexity for service delivery teams.

Managing offered services and integrating with customer catalogs

From the perspective of the service delivery catalog, accounts are managed by managing the life cycle of offered services associated with the account. The responsibility for offered services is fully within the scope of the account manager, as ***Figure 5*** shows.

In addition to its use in managing component life cycles, the set of offered services of an account is published to a customer catalog from which customer employees can order services. Frequently, customers choose to maintain their own service catalog items either in whole or in part. In this case, the SDMP requestor must be able to access the

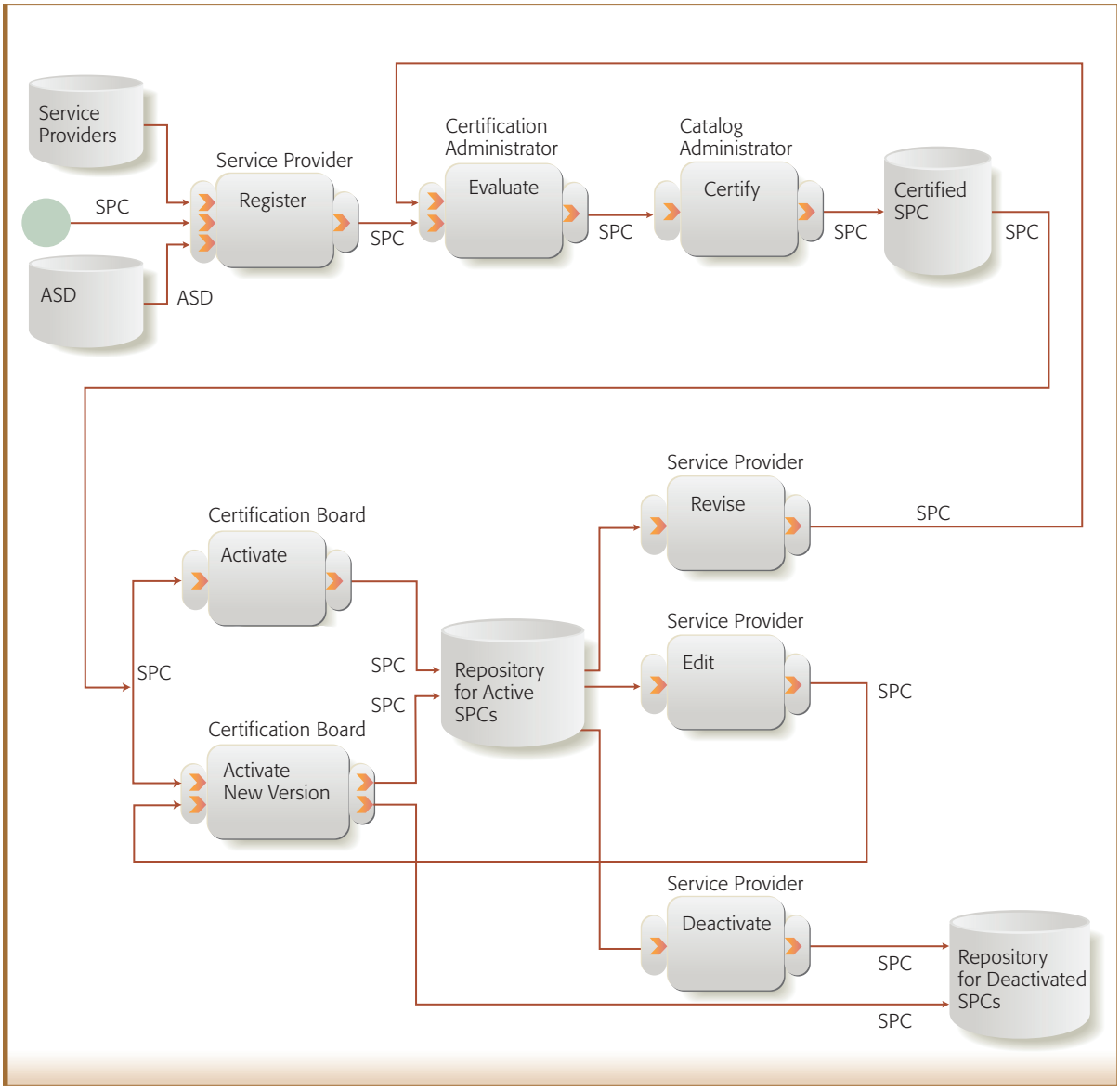

Figure 4
Life cycle of SPCs

customer's catalog during the ordering process. There are three ways to provide access to the customer's catalog: the link-out, punch-out, or full-integration functions.

Of the three methods of accessing the customer's catalog, the link-out function is the simplest. Consisting of a URL (uniform resource locator) link to the customer's system, the user clicking on the link is brought to the new site where the order is placed. The order is fully independent of the SDMP, with no connection between the systems at all. The punch-out function is more complex, allowing the user to choose an item in the SDMP but have fulfillment of the service in the customer's system. When the item to order is located and selected, the user is brought seamlessly to the customer's catalog where the item is configured and ordered. The ordering activity is transparent to the user. The full-integration function is the most complex. This technique allows the customer catalog to appear within the SDMP. Full integration requires customers to maintain their catalogs with the same data

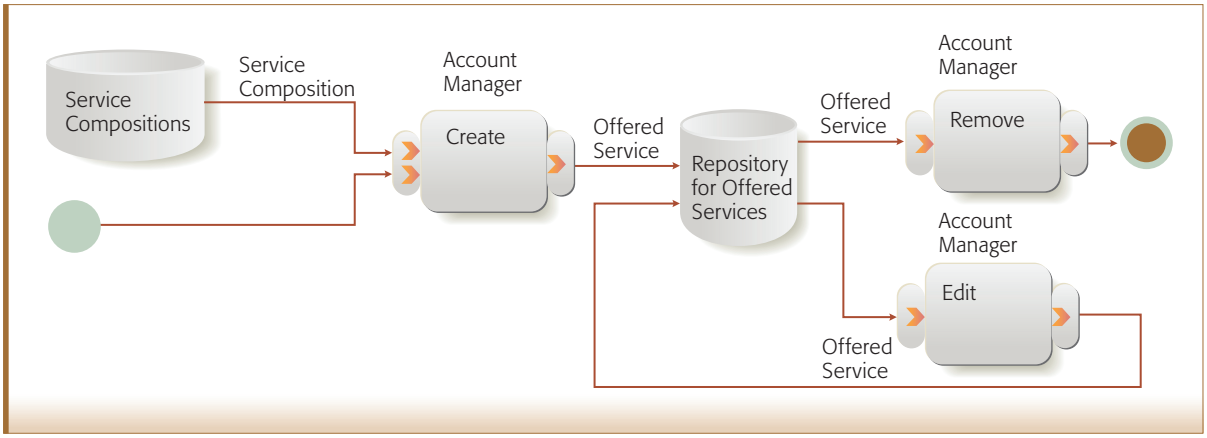

Figure 5
Life cycle of offered services

structures as the SDMP or to use a transformation program to adapt the catalog to it.

IMPLEMENTATION

To create a usable catalog management system, it is not sufficient to manage catalog elements according to predetermined rules as described previously. To be usable, the system must also help guide the user through the process of creating and editing catalog elements. To address this usability issue in managing the SDMP service delivery catalog, the catalog management implementation is divided into a data management component and a user interface component. The data management component manages communication with the underlying databases, including authorization, role access, and life-cycle management. The user interface component implements an editing tool that guides the user in the process of creating and maintaining catalog content.

Data management

The data management component is required to support multiple roles and multiple data types (atomic services, composed services, offered services, providers, etc.) with different life-cycle requirements. This complexity was naturally handled by treating content management for each data type as a distinct business process. By doing so, we were able to leverage existing tools for developing formal models for business processes[4] and transforming these formal models into a software implementation based on the model-driven business transformation framework. The resulting software implementation executes as a server application under WebSphere Process Server[5] Version 6.0.1, with a corresponding client communicating with it through RMI (remote method invocation).[6]

Model-driven business transformation

Model-driven business transformation (MDBT) is an innovative framework for flexible and efficient realization of business components on an IT platform.[7] MDBT leverages concepts from SOA and Model-Driven Architecture (MDA**). The framework is made up of four layers: strategy, operations, execution, and 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 component. The operation layer describes the operations performed by the business to achieve these goals. The execution 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.

Operation models present the perspective of the line-of-business managers. They constitute the abstractions needed to describe the business operations that the business employs to achieve the business goals. Business operations are modeled as a factorization of the operational knowledge into two parts, business tasks and artifact repositories. A business

operation is represented as a connected graph with the business tasks, artifact repositories, and other business operations as nodes and the flow of artifacts between these elements as edges. Business operation models for the SDMP catalog components are shown in Figures 3 and 4.

The key abstraction in the execution layer is the concept of an adaptive business object (ABO).[8] An ABO is the execution-level abstraction of a business artifact. It models the structure and behavior of the artifact. The key elements of the ABO are as follows. The life cycle of the business artifact is defined by using a finite state machine.The states of the finite state machine correspond to the life-cycle states of the artifact. An ABO receives external events through its public interface and reacts to these events with action invocations triggered as part of the state transitions. An ABO uses a data graph to dynamically aggregate information on demand from heterogeneous data sources. People and applications may interact with the ABO at various points in its life cycle. The modeler can specify read, write, or search access for data and authorize access for events for each business role and state combination. The data actions are used to model CRUD (create, retrieve, update, and delete) operations on parts of or the whole data graph. These are invoked by the ABO as a side effect of state transitions. An ABO changes its environment through remote actions which are triggered as part of the state transitions. Views present the external interface or API (application programming interface) of the ABO.

The catalog solution is composed of five ABOs: atomic service, service composition, offered service, service provider, and SPC. The user interface component of the catalog, discussed in the next section, interacts with these ABOs by using the view interface.

User interface

The user interface component is implemented as a dynamic Web server application using a standard model-view-controller paradigm.[9] The user interface will run on any application server capable of supporting J2EE** Version 1.4[10]; for simplicity, we chose the WebSphere Process Server. The user interface makes use of Jakarta Commons[11] utilities and XML parsers developed using XML Beans.[12] Views are implemented as JavaServer Pages** (JSPs**) that communicate with the business logic through Java** Beans, and the controller is implemented using Java Servlets**, both of which are part of the J2EE specification.

The model separates data manipulation from business logic. Business logic is handled by action Java Beans, one corresponding to each JSP page. Data manipulation is handled by data wrappers, one corresponding to each underlying data type (atomic service, service composition, etc.). Data wrappers hide the underlying data management client API from the business logic. The business logic only manipulates wrappers as Java interfaces; the binding between wrapper and wrapper implementation is mediated by a class that is initialized at runtime. Upgrades to the data management client API provided by MDBT require only changes to the wrapper implementation classes and not to the business logic itself; this has proved useful in permitting data management and business logic to be developed in parallel.

ALIGNING BUSINESS GOALS USING THE SERVICE DELIVERY CATALOG

Implementation of the service delivery catalog must closely align with the desired business goals of both the recipient of the services and the provider. Because the catalog defines how services are requested and delivered in a standard framework, there is an opportunity for both parties to align services closely with demand, evaluate service delivery costs, prevent SLA violations, and create accurate records for billing purposes. This section outlines how services are mapped to the provider portfolio and how the catalog assists in ensuring that business goals are met during runtime and over a variable accounting period.

In the case of most service providers, services as they are sold to a customer come from a different portfolio than services as they are actually delivered. In the SDMP methodology, the portfolio of services sold is mapped back to the portfolio of services delivered. From that, best practices are leveraged to create the delivery catalog based on the specific requirements of the customer. As shown on the left side of *Figure 6*, services are sold to the customer based on delivery themes such as infrastructure access services. This theme is very general and maps to one through many service elements in the delivery portfolio in the middle of the figure. In this example, infrastructure access services map to

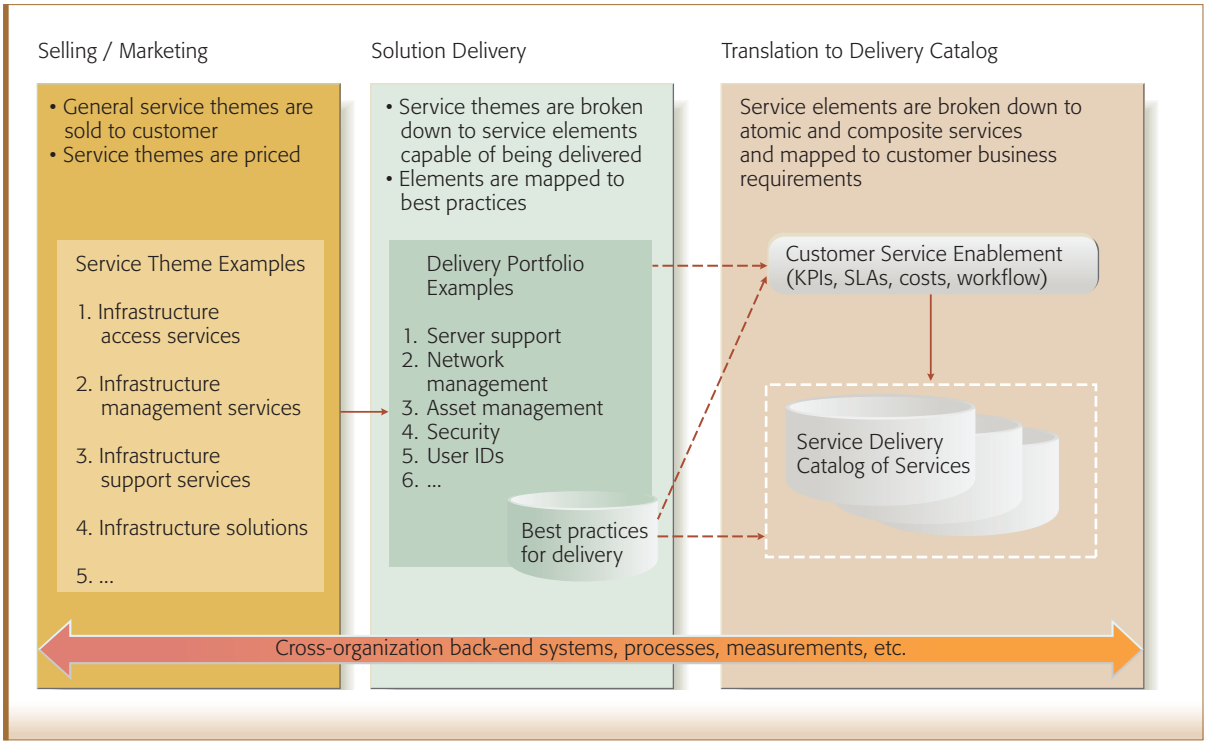

Figure 6
Mapping of services as sold to atomic and composite services

server support, network management, and so forth. In turn, each of these service elements maps to best practices, which, in turn, map to actual service-delivery atomic and composite services based on customer business requirements. Once the services for delivery are created in the catalog, they can be made available for the end user to request.

At runtime, service orders are tracked and managed in the context of the business objectives. Probably the most common business objective is ensuring that SLAs for delivery times are met. The SDMP can accomplish this at the task, atomic, or composite levels. SLAs can be set for a type of service, such as e-mail services, or for any logical organization, such as a department. Once the SLAs are set, thresholds for notifications can be set. The tracking of rate and total consumption of services are also important. In the runtime tool, entitlements that have been agreed upon can be specified, and orders can be counted and compared with the entitlements. Once a specified entitlement is met, the tool can either prevent a new order or warn the requestor that another order will incur additional cost. Over a long period of time, catalog data can be gathered for billing purposes or mined to set baselines and re-engineer services.

Baseline data for common KPIs such as average turnaround time and average cost per order or service type can be gathered from the data repository. The KPI data can be used to compare similar services, both within a customer's delivery environment and, if the provider handles multiple customers, between customers. Comparisons with industry standards are also possible, although these tend to be less meaningful, given the lack of standard service definitions.

Using this comparison data, substandard services can be identified and targeted for re-engineering efforts. Re-engineering efforts are facilitated by utilization of composite-, atomic-service-, or even task-level comparison data. Inefficiencies can thus be pinpointed and root cause analysis performed. Recurring problems due to a particular performer or task complexity can be addressed by re-engineering. The re-engineered service, once in place, can be measured and further revised based on the results. Thus, feedback loops are created and used. Finally,

the improved results can be communicated to marketing for revising the pricing offered to customers.

RELATED WORK

The SDMP service delivery catalog touches upon multiple fields of related work. There are multiple approaches in representing meta-information on services, in particular in the field of Web services. UDDI (Universal Description, Discovery, and Integration) provides a directory service representing metadata on services for search and binding.[13] WS-Policy and derived standards such as WS-Security can enable representation of rich information on services.[14] Approaches related to the semantic Web such as OWL-S[15] (Ontology Web Language for Services) provide the means to represent properties of services based on description logic, including process-related information.[16] WS-BPEL, which is also used in the approach presented here, is a standard means to describe process-type compositions of Web services.[2]

Service catalogs are recognized as a central structure of the IT Infrastructure Library** (ITIL**). However, neither ITIL, nor any other standards bodies, define how service catalogs are to be implemented or managed. Individual IT organizations have been left on their own to create their own versions. Various ITIL service management consultants such as Pink Elephant[17] communicate and inform the IT industry about service catalogs. However, the focus is on information and not on a standard architecture or management model.

Several software vendors have products that have some aspect of SDM. There are vendors whose primary capability is SDM (e.g., newScale, Inc. or MRO Sofware, Inc., which was recently acquired by IBM). There are also vendors whose products have incorporated some aspect of SDM, either a catalog or a workflow (e.g., BMC** Remedy**). Finally, some vendors' products enable IT governance or IT project management, or both (e.g., Computer Associates' Clarity[18] or the IBM Rational* Portfolio Manager).[19]

Although all of these approaches cover aspects of the issues handled by the service delivery catalog, none of them provides a model for representing the business-level information in conjunction with the technical details of services as outlined here. Furthermore, the catalog items and associated delivery processes lack the life-cycle management and composition approach necessary to support the requirements of a service provider.

SUMMARY, CONCLUSION, AND FUTURE WORK

The SDMP addresses the requirements of large-scale service delivery, in particular the management and fulfillment of service requests. This includes dealing with a large number of customers and their specific requirements as well as large service delivery organizations whose teams are located around the world at global service delivery centers and external service providers. The complexity and variation of service delivery processes are dealt with by use of a component-centric approach that is based on atomic services, specialized service subprocesses assigned to a team, and service compositions that describe a workflow in which a set of atomic services is invoked to fulfill a service request from a customer.

The service delivery catalog is the tactical management instrument for the SDMP. It provides information on service compositions and atomic services to the runtime system. It manages subscriptions of services and their customization and SPCs that deliver atomic services for accounts. The service delivery catalog implements the governance processes for all artifacts in the catalog to guarantee smooth initiation of new accounts, enlisting of service providers, and changes to the service designs in new versions.

Based on the component-centric approach, using ASDs and service compositions, the SDMP enables a service delivery organization to scale and work productively. As a primary result, processes are not run in an account-specific manner by account-specific teams. Instead, service delivery teams work in the context of atomic services that can be reused in different compositions. Compositions can be shared between accounts but can be customized to specific account requirements if necessary. Shared service compositions also enable benchmarking of services among accounts. Finally, service delivery teams are, to some extent, insulated from changes in service compositions by adhering to ASDs as interfaces and, hence, can improve their productivity.

Despite these improvements, there are still many remaining issues. The transition of new accounts into a component-centric SDMP service delivery

environment is costly, and the initial investment must be compensated for by lower costs when services are delivered. We typically encounter proprietary tooling and complex ("spaghetti") integration of Web sites, Lotus Notes* databases, e-mail conventions, and spreadsheets that enable service processes for specific accounts, which must be disentangled and transformed to fit the service component structure of atomic services.

In the future, we plan to integrate the SDMP approach with the IBM Tivoli Service Management product line based on the MRO Maximo** platform.

*Trademark, service mark, or registered trademark of International Business Machines Corporation in the United States, other countries, or both.

**Trademark, service mark, or registered trademark of Object Management Group, Sun Microsystems, Inc., United Kingdom Office of Government Commerce, BMC Software, Inc., or MRO Software, Inc. in the United States, other countries, or both.

CITED REFERENCES

1. S. Kumaran, P. Bishop, T. Chao, P. Dhoolia, P. Jain, R. Jaluka, H. Ludwig, A. Moyer, and A. Nigam, "Using a Model-driven Transformational Approach and Service-oriented Architecture for Service Delivery Management," *IBM Systems Journal* **46**, No. 3, ***–*** (2007, this issue).
2. *Web Services Business Process Execution Language Version 2.0, Committee Specification*, A. Alves, A. Arkin, S. Askary, C. Barreto, B. Bloch, F. Curbera, M. Ford, et al. (Editors) (January 31, 2007), http://docs.oasis-open.org/wsbpel/2.0/CS01/wsbpel-v2.0-CS01.doc.
3. J. Sermersheim, *Lightweight Directory Access Protocol (LDAP): The Protocol (RFC 4511)*, The Internet Society (June 2006), http://tools.ietf.org/html/rfc4511.
4. WebSphere Business Modeler—Family Overview, IBM Corporation, http://www.ibm.com/software/integration/wbimodeler/.
5. WebSphere Process Server—Product Overview, IBM Corporation, http://www.ibm.com/software/integration/wps/.
6. *Java Remote Method Invocation (Java RMI)*, Sun Microsystems (2004), http://java.sun.com/j2se/1.5.0/docs/guide/rmi/index.html.
7. S. Kumaran, "Model-Driven Enterprise," *Proceedings of the Global Enterprise Application Integration Summit*, Banf, Canada (2004), pp. 166–180.
8. P. Nandi and S. Kumaran, "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.
9. F. Buschmann, R. Meunier, H. Rohnert, P. Sommerland, and M. Stal, *Pattern-Oriented Software Architecture: A System of Patterns*, John Wiley & Sons, Hoboken, NJ (1996).
10. Java 2 Platform, Enterprise Edition (J2EE) 1.4, Sun Microsystems (2006), http://java.sun.com/j2ee/1.4/.
11. Jakarta Commons, Apache Jakarta Project, http://jakarta.apache.org/commons/.
12. The Apache XML Project—Welcome to XMLBeans, http://xmlbeans.apache.org/.
13. UDDI Version 3 Specifications, OASIS (2005), http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm#uddiv3.
14. *Web Services Policy 1.2 - Framework (WS-Policy)*, S. Bajaj, D. Box, D. Chappell, F. Curbera, G. Daniels, P. Hallam-Baker, M. Hondo, et al. (Editors), W3C Member Submission (April 2006), http://www.w3.org/Submission/WS-Policy/.
15. *OWL-S: Semantic Markup for Web Services*, D. Martin, Editor, W3C Member Submission (November 2004), http://www.w3.org/Submission/OWL-S/.
16. OWL-S 1.1 Release, http://www.daml.org/services/owl-s/1.1/.
17. Pink Elephant, http://www.pinkelephant.com.
18. Portfolio and Financial Management, http://ca.com/clarity.
19. Rational Portfolio Manager—Product Overview, IBM Corporation, http://www.ibm.com/software/awdtools/portfolio/index.html.

Accepted for publication February 23, 2007.
Published online July 2, 2007.

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/.

John Hogan
IBM Integrated Technology Delivery Division, 2455 South Road, Poughkeepsie, NY 12601 (jphogan@us.ibm.com). Mr. Hogan is a senior software engineer and currently works in the Integrated Technology Delivery division. He has a B.S. degree in finance from Central Connecticut State University and an M.S. degree in management information systems from the University of Arizona. During his career at IBM, Mr. Hogan has designed and deployed IT systems management solutions for a variety of corporate customers. His current interests include using service catalogs and workflow to improve the efficiency of IT service delivery.

Rajesh Jaluka
IBM Integrated Technology Delivery 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 service industries.

David Loewenstern
IBM Research Division, Thomas J. Watson Research Center, 19 Skyline Drive, Hawthorne, NY 10532 (davidloe@us.ibm.com). Dr. Loewenstern is an advisory software engineer in the Service Delivery and Networking Services Research department of the Watson Research Center at Hawthorne. Before joining IBM in 2006, he had been active as a researcher in artificial intelligence, as a computer scientist at Bell Laboratories, and as a software consultant. He received his Ph.D. degree in computer science from Rutgers University in 1999.

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.

Allen Gilbert
IBM Software Group, Tivoli, 11501 Burnet Road, Austin TX 78758 (amgilber@us.ibm.com). Mr. Gilbert is a senior technical staff member at the Tivoli development laboratory in Austin, Texas. He is a lead architect on the Tivoli IBM Service Management (ISM) development team, and is currently leading the design of the ISM Service Catalog and Service Desk products. Before this, he led the development of the Policy Management component of the Tivoli Autonomic Computing initiative. Mr. Gilbert joined IBM in 1990 as an OS/2® architect. Prior to that, he held a variety of technical and management positions while developing operating systems for Data General and Control Data Corporation and building robotic and factory automation software systems for the American Cimflex Corporation. He has a B.S. degree in biology from SUNY at Stonybrook and an M.S. degree in genetics from the University of California at Davis.

Arijit Roy
IBM Global Business Services, IBM India, Millennium City, Block DN, Sector V, Saltlake, Kolkata WB 700091 (arijiroy@in.ibm.com). Mr. Roy is a senior system engineer in IBM India. He has a bachelor's degree in computer science and engineering from the National Institute of Technology in Durgapur, India. He joined IBM India in 2005. A member of the core MDBT/BPM (Model Driven Business Transformation/Business Performance Monitoring) technical group, Mr. Roy is the leader of the MDBT Practice Team in IBM India. He is also a certified WebSphere Application Server professional and has given technical training on Web Services in IBM India. In addition, he has contributed to the development of SOMA (Service-Oriented Modeling and Architecture) and cultivated the growth of SOA in IBM India. Since joining IBM, he received the Bravo Award in 2005 and the Key Talent Award in 2006.

Thirumal R Nellutla
IBM Integrated Technology Delivery Division, 10 N. Martingale Rd, Schaumburg IL 60173 (thiru@us.ibm.com). Mr. Nellutla has been a lead architect in the Integrated Technology Delivery division since 1997. He has held a diverse number of technical leadership roles in the areas of systems architecture, infrastructure architecture, and all disciplines of IT management. Before joining IBM, Mr. Nellutla worked at prestigious institutions in India and significantly contributed to the development of distributed-architecture-based parallel processing systems. His primary areas of focus include IT operations, optimization, and IT systems and services management in support of the IBM strategic outsourcing and e-business hosting lines of business. He has been a recognized leader in the creation of Web load-balancing architectures and the creation of several service-offering capabilities in support of the service delivery business. He contributed significantly to the evolution of the e-business on demand architecture and associated components of the IT utility model. His other interests include the impact of technology adoption on human society.

Maheswaran Surendra
IBM Research Division, Thomas J. Watson Research Center, 19 Skyline Drive, Hawthorne, NY 10532 (maheshsurendran@in.ibm.com). Dr. Surendra received a Ph.D. degree in chemical engineering in 1991 from the University of California at Berkeley and has been at IBM Research since then. He has worked in technical areas ranging from semiconductor manufacturing to software systems management, and most recently in IT service delivery. He is currently a senior manager in the Services organization in IBM Research, and his focus is the application of IT service management technologies in service delivery operations. ■