IBM Service Management architecture

D. Lindquist
H. Madduri
C. J. Paul
B. Rajaraman

In this paper we describe the IBM Service Management (ISM) architecture, a service-oriented architecture designed to automate and simplify the management of business services. We describe the four major components of ISM: portal-based user interfaces, a process layer that includes the process runtime and service management solutions, an information layer that includes a configuration management database, and operational management products and their integration with service management processes. We describe the way in which the service management solutions are based on industry best practices, and in particular ITIL® (Information Technology Infrastructure Library®). We discuss our experiences implementing ISM and conclude with ideas for future work, including how ISM lays out the groundwork for the future implementation of autonomic functions.

INTRODUCTION

Many information technology (IT) organizations are asked to reduce costs and improve the quality of service at the same time that the complexity and the change and compliance requirements are accelerating. The traditional approach to systems management, which tends to create organizational "silos" of expertise for specific management domains, no longer meets these demands.

A number of studies indicate that nearly 70 percent of chief-information-officer (CIO) budgets cover labor costs, and of these, more than half are expended on operations. The efficiency of traditional IT operations appears to be waning. The root cause of this inefficiency is most likely complexity. Enterprise application configurations frequently evolve into haphazard interconnections of accumulated software and hardware that look like complex wiring diagrams. It is not uncommon for an enterprise to have dozens of applications based on different application architectures. As these complex topologies grow, operations teams are addressing complexity with the help of advanced resource-oriented tools, which leads to islands of expertise: server experts, application experts, storage experts, network experts, security experts, and so on. Although effective within each domain, this approach is not effective in managing complex application environments.

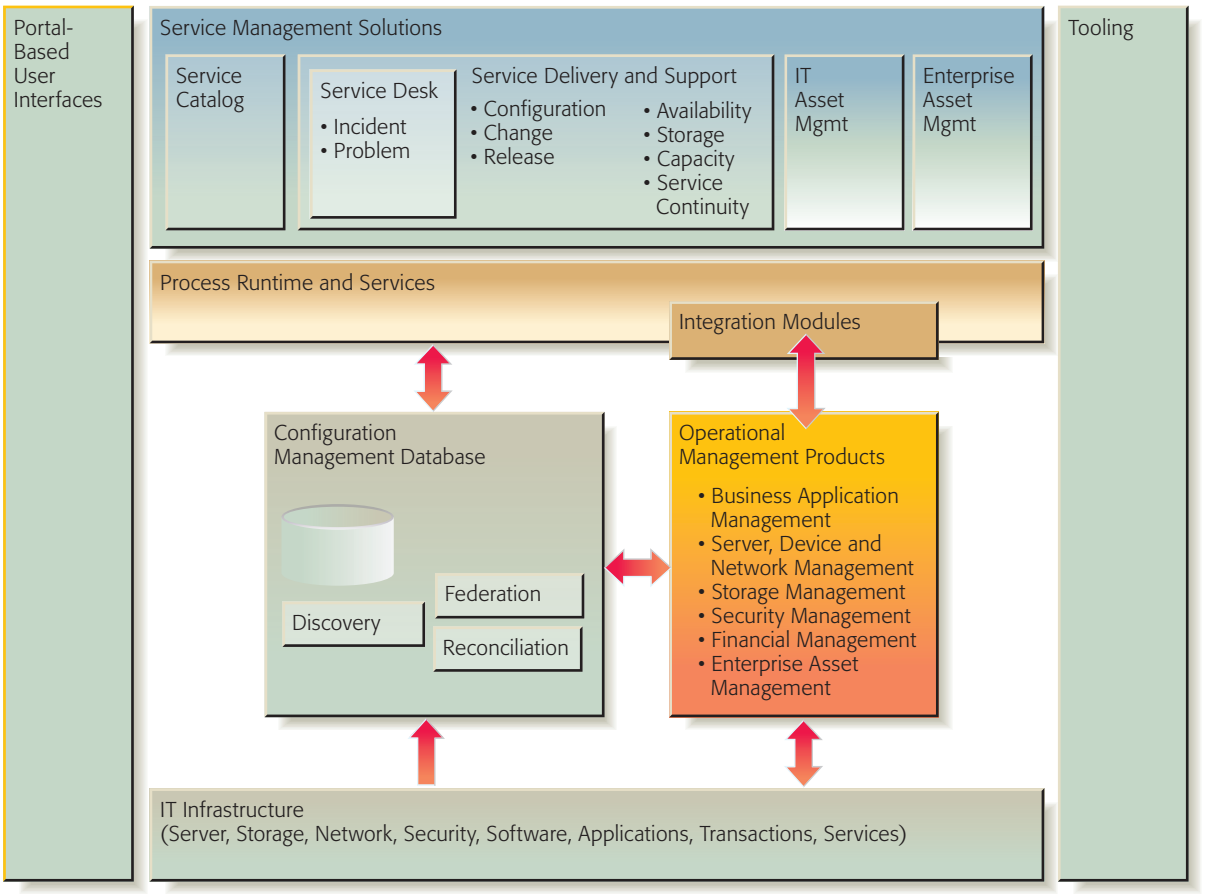

Figure 1
IBM Service Management architecture

In summary:

- Traditional systems management processes are fragmented by resource domain (networks, storage, servers, applications, databases, etc.), and there are few tools that integrate these processes.
- The effectiveness of IT processes depends upon authoritative configuration data, but these data rarely exist. Consequently, it is difficult to generate a comprehensive and accurate picture of the state and configuration of the IT environment.
- The various systems management products are not integrated, which leads to inefficient manual tasks. Because user interfaces (UIs) and data models are not integrated, the opportunities for automation are limited and the use of best practices in IT governance is impeded.

IBM Service Management (ISM) is an approach designed to automate and simplify the management of business services. In this paper we describe the ISM architecture, a service-oriented architecture (SOA) that builds upon tools, techniques, and architectures for business-process implementation and transformation, information management technologies, and the breadth of the industry operational-management technologies.

The ISM architecture illustrated in ***Figure 1*** is comprised of four major components: (1) a user interface that represents a portal-based integration of UIs for user interactions and collaboration (labeled Portal-Based User Interfaces), (2) a process layer that includes the entity labeled Process Runtime and Services and the entity labeled Service Management Solutions, (3) an information layer represented by the configuration management database (CMDB), and (4) an operational-management-technologies component represented by the entities labeled operational management products (OMPs) and the adjacent integration modules.

Figure 1 also shows Tooling, a collection of tools to create and modify processes, UIs, and data. The ISM architecture consolidates current IT components and management functions along several dimensions. All user interactions are consolidated at the portal; the roles and responsibilities of the users are integrated with the defined processes; and activities (subprocesses) are integrated with configuration data from the information layer. The architecture takes advantages of software middleware and industry standards for portal, workflow, and data federation.

The ISM architecture enables client IT process transformation based on business process workflow tools, information integration technology, and operational management technology. Through many engagements with clients and our service teams, we recognized that to address the increasing complexity of IT, we needed to focus on both the IT processes used by clients and the automation of manual tasks. The architecture provides a way for clients to transform their existing processes to incorporate best practices and gradually automate processes such as provisioning, orchestration, and problem determination. We also realized that the effectiveness of these IT processes depended upon access to accurate information that described the authorized and discovered states of the IT resources, commonly known as configuration management data. The service management solutions we developed are based on IBM and industry best practices, such as the Information Technology Infrastructure Library** (ITIL**),[1,2] Control Objectives for Information and related Technology (COBIT**),[3] and Enhanced Telecom Operations Map** (eTOM**).[4]

The process layer hosts solutions based on the concept of service management processes (also referred to as Process Managers or PMs). The service management processes are integrated with operational management technologies and the CMDB. The service management processes and related tooling incorporate a set of best practices that may be modeled and customized to support existing processes. Selected tasks within these processes may be progressively automated through operational management tools, directly reducing IT management costs in a manner consistent with organizational responsibilities. The integration of these tasks with the systems management technology (implemented through OMPs) is accomplished through the use of an SOA.[5]

The information layer, which includes the federated CMDB, provides automated application discovery and detailed views of system, software, and service topologies. Open interfaces provide ease of integration with process, data sources, and automation technology. Information about IT resources, topology, and relationships is often dispersed throughout operational registries used by management tools; without including this information in a federated database, a logical view of all the IT resources and their respective relationships and dependencies is not available. This logical view is critical to improving the efficiency and effectiveness of processes. For example, to understand the impact of a change request and to implement a successful change management process, information about the current state of resources, the relationships to business applications, the service level objectives, the compliance policies, and the dependency on other resources are all critical information aspects. The integration of the service management processes and the CMDB with operational management technologies forms the core of our ISM architecture.

The rest of the paper is organized as follows. In the next section we describe the processes involved in a typical scenario, in this case the deployment of a software upgrade, and we derive a set of requirements for our architecture. In the following section we describe the ISM architecture and the way it addresses the requirements. In the next four sections we focus on four major aspects of the ISM architecture: (1) the CMDB, (2) the OMPs and their integration with service management processes, (3) service management processes, including the use of a service catalog for requesting services, and (4) the role of service management processes and implementation considerations for creating processes. The next-to-last section describes our experiences implementing ISM. The last section contains a summary and ideas for future work.

HIGH-LEVEL REQUIREMENTS FOR THE ISM ARCHITECTURE

In this section, we describe the high-level requirements for the ISM architecture, beginning with an end-to-end scenario involving a request for a software upgrade. We then identify requirements

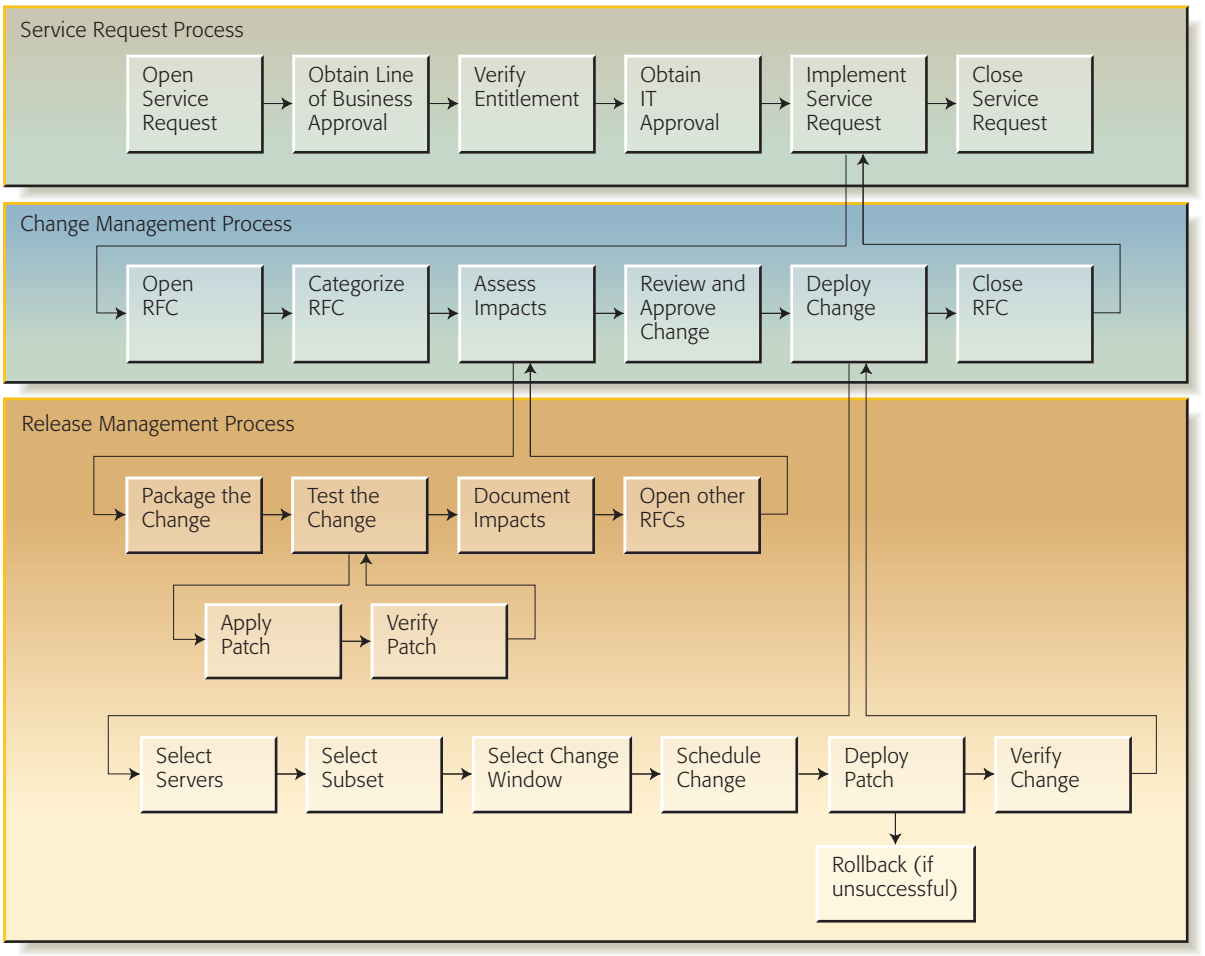

Figure 2
Change-management and release-management activities for deploying a software upgrade

associated with the architectural components and their interfaces.

Let us consider the case of a centralized IT department within a large enterprise whose end users are grouped within organizations (the various lines of business within the enterprise). The IT department provides various services to the end users which are accessed through a service catalog. The services in the service catalog are triggered by a service request process that may include approvals from the organization (line of business), procurement of resources, and other necessary activities.

One of the services offered by the IT department is an application software upgrade for a *production* environment. Production environments involve servers running mission-critical applications and are typically governed by a rigorous change process intended to minimize the risk of disruption and to ensure that the appropriate authorizations have been obtained, the stakeholders notified, and the configuration information correctly updated.

Figure 2 shows the steps in the service request, change, and release-management processes for deploying a software upgrade in a production environment. The end user submits a service request through a service catalog. The service request process defines how the service request is handled. The service request process starts with approval by the line of business, followed by verification that the user is entitled to the service, verification that the IT department can provide the service with the required parameters (such as the date by which the upgrade should be completed), and finally, the implementation of the infrastructure changes specified by the service request; that is, the

application upgrade requested by the end user. Additionally there may be other related changes that are necessary to implement the upgrade (such as a new version of a monitoring agent that should be deployed).

The infrastructure changes specified by the service request are typically managed through IT service management (ITSM) processes, specifically, the ITSM change-management and release-management processes. Best practices for these processes are defined at a high level by ITIL. A request for change (RFC) is first opened that specifies the software upgrade. The RFC information is stored as a *process artifact* in the CMDB. For example, the RFC specifies the application to be upgraded, the software release levels, and so on. The categorize-RFC step can be performed either by a program or by a person. The categorization may depend on the importance of the upgrade based on business value as well as the level of service guaranteed to the end user. Once the change is accepted, the impact of the change is assessed. Software upgrades require testing to ensure stability and performance. Impact assessment involves steps in the release management process: package the upgrade, test the upgrade, and open RFCs to address other impacted systems (for example the application monitoring system). There may be additional considerations involved in impact assessment, such as maximizing the availability of the application. The result of impact assessment may require that the RFC also be handled by other service management processes, such as availability management and service continuity management.

Following impact assessment the change is reviewed (Review and Approve Change). If approved, the application upgrade request is scheduled for deployment. The deployment of the upgrade also involves steps in the release management process for identifying and staging the target systems as well as performing the actual upgrade of the application. The deployment of software upgrades can be accomplished through the use of OMPs such as Tivoli* Provisioning Manager,[6] which automates the distribution and installation of the software upgrade and ensures that the upgrades are performed consistently across multiple systems. The last step in the deployment involves verification of the change, following which the RFC is closed.

This scenario illustrates the need to integrate (data as well as interfaces) the change management process, the release management process, and the OMPs. The CMDB plays a crucial role by (1) providing a repository of resource and process information to tasks, (2) providing a consistent data representation across processes, and (3) enabling the integration of OMPs with processes. For example, the relationship between resources maintained by the CMDB can be used to assess the business services impacted by a software upgrade to a specific application component of a business service.

The ITSM process described in Figure 2 defines the high-level *activities* in the change-management and release management processes. Within each of these activities the IT staff (users) perform *tasks* that implement the specified activities. Users perform these tasks in the context of a role and authority associated with the role. For example, testing the software upgrade may first involve the task of checking out a build from a development repository, followed by tasks to define the test case, approvals to ensure test coverage, provisioning a test environment, and so forth.

As organizations evolve to adopt best practice processes, standardization typically occurs at the level of process activities. However, the specific tasks underlying each activity are often specific to the organization. Additionally, these tasks change at a much higher rate based on implementation considerations, such as types of changes managed by the process (e.g., software upgrades vs storage changes), geographical considerations, and evolution of technology. For example, tasks performed for impact analysis in virtualized infrastructures can be significantly different from environments with dedicated servers. The ISM architecture allows the definition and extensibility of tasks (and associated UIs and data) through tools that reflect the skills of IT staff configuring the processes for use in their organizations. The relationship between process activities and tasks is shown in ***Figure 3***.

In Figure 3, the change process is represented by the standard activities of open RFC, categorize, assess, approve, deploy, and close RFC. Some of the key tasks in the deploy activity are also shown. For example, deployment of a change requires the assignment of personnel and creation of the change package (tasks 1 and 2, which are often manual activities) and the actual deployment of the change (task 3, which is often implemented through

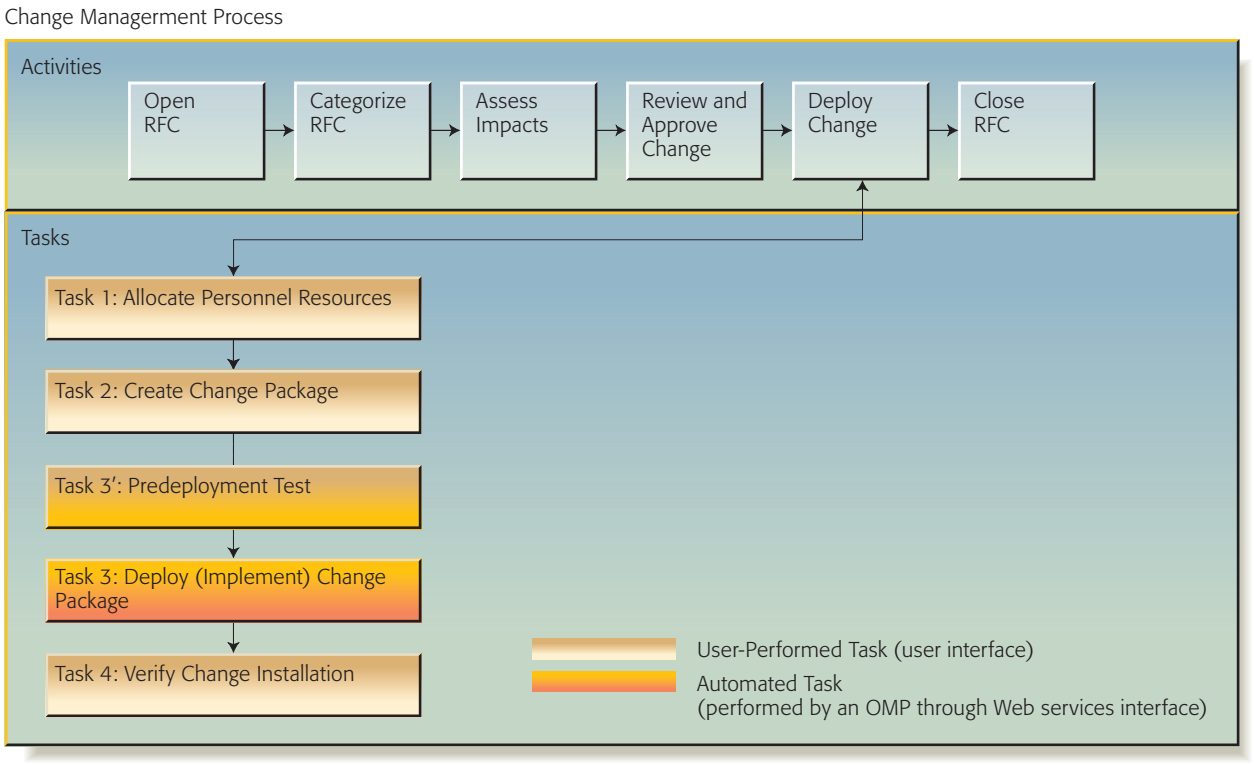

Figure 3
Selected activities and tasks in change management

automation tools such as the Tivoli Provisioning Manager for Software). For applications likely to incur deployment errors, a task that does a predeployment test should be added. The ability to easily add such a task (task 3′ in Figure 3) is a critical requirement for the IBM Service Management architecture.

From the analysis of the preceding scenario, we can define the following high-level requirements for the ISM architecture:

1. Provide best-practice processes, including activities and tasks and definitions of the associated roles.
2. Provide a consistent and integrated operational environment (runtime) in which to define and execute processes as well as integrate different processes.
3. Provide capabilities for collaboration and coordination of activities and tasks across multiple roles with secure access control (authentication and authorization).
4. Provide a CMDB to store resource data, process artifacts, and relationships between them. The CMDB should be closely integrated with the process runtime in order to enable both end users and automated tasks to represent and use data consistently.
5. Integrate service management processes with OMPs to support automation of management operations.
6. Provide UIs that enable the aggregation of information and views from the CMDB and OMPs.
7. Support ease of use in creating and configuring process tasks (including extensible UIs, logic, and data) to address the intrinsic variability of tasks associated with customized implementations. Additionally, the architecture must support integration of processes with IBM and third-party OMPs as well as other business and IT processes.

ISM architecture

In this section we elaborate on the high-level requirements identified in the prior section, focusing

on the primary capabilities needed to realize an implementation of the architecture.

Best practices for service management processes

The ISM architecture allows the implementation of best practice processes (e.g., ITIL, COBIT, and eTOM) for various domains of service management, such as change management, release management, incident management, and problem management. These best practices are captured in IBM Tivoli Unified Process (ITUP),[7] which in addition to documenting these best practices also provides a tool for planning and implementing them. ITUP can be used by process designers to adopt and implement service-management best practices by using the ISM architecture.

Consistent and integrated operational runtime for process execution

The ISM architecture supports the coordination and automation of tasks and collaboration between users in multiple roles. For example, the creation of an RFC is initiated by a user in the role of change requestor while the acceptance and classification of the RFC is performed by a user in the role of change reviewer or change manager. In addition, processes addressing a particular service management domain may need to interact with processes (and roles) in other domains.

Processes, activities, and tasks may be completely manual (i.e., for a given process activity, the user interacts with the process through a UI and performs required tasks manually), completely automated (i.e., after initiation, the tasks defined by each process activity are carried out without human intervention), or partially automated (i.e., certain process activities or tasks are performed manually, and others may be automated).

The ISM runtime architecture provides the capability to define consistent and repeatable high-level processes while accommodating the variability and ease of configuration of tasks underlying process activities. This is achieved by leveraging the standard middleware runtime capabilities of IBM WebSphere* Business Integration and the Web Services Business Process Execution Language (WS-BPEL) for formal process definition, runtime, and monitoring and integrating these with task and work-management capabilities and associated tooling.

Coordination across multiple roles with secure access control

A key feature of the ISM implementation is the support for users (IT staff) to view, claim, reassign, delegate, and perform tasks. Additionally the implementation provides integrated notification through standardized and parameterized communication templates. UIs, applications, and an integrated security model allow a user to view and claim tasks that are assigned to the specific user or to the role the user performs in the process. In addition, the security model also provides authorized access to objects referenced by the task. Escalations can be defined on the state of a process task, which allows for actions (e.g., notification) to be taken when, for example, a task is not completed within a prescribed amount of time. Future enhancements include deeper integration with collaboration capabilities such as instant messaging.

Configuration management database

ISM processes and tasks primarily deal with managing resources across various inter-related resource and management domains. The ability to discover resources, their configuration, and their relationships with other resources are core capabilities for implementing service management processes. Modeling and storing resource-configuration and relationship information allows the establishment of processes to control configuration changes to resources. This is critical to all other service-support and delivery processes. The CMDB is the repository that maintains configuration and relationship information about resources.

In the ISM architecture, the CMDB has the following key capabilities:

1. *Discovery, application mapping, and visualization*—This capability discovers resources and relates the resources to the business applications and services that depend on them. Discovery can be targeted directly against resources in the IT infrastructure, against information about the resources gathered by other management systems or against data manually maintained by end users (e.g., spreadsheets). In addition, capabilities are provided to relate resources to the business applications that they support. This ranges from

automated mapping based on resource configuration information to manually specified relationships.

2. *Federation*—Maintaining all possible resource-configuration and relationship information in the CMDB can result in scalability issues as well as the overhead in maintaining consistency of changes. A federated approach allows the ability to access information about the resource configuration by accessing other management systems that have more detailed information about the resource. Federation must support various data formats and repositories, including relational databases, XML (Extensible Markup Language) documents, spreadsheets, and document repositories.
3. *Reconciliation*—A resource may be managed by multiple management systems, each of which might be responsible for a particular aspect of management (e.g., IBM Tivoli Monitoring may monitor the resource, whereas IBM Tivoli Provisioning Manager for Software may be responsible for software distribution to the resource). It is necessary to reconcile the various methods by which a resource is identified by individual management systems so that a single instance of the resource exists in the CMDB. This is accomplished by allowing one or more naming rules to be defined. A prioritized set of naming rules allows the reconciliation of multiple internal naming schemes to be recognized, given that one or more attributes which are required by the naming rules are made accessible by the individual management systems. Naming rules are described in detail in a companion paper on the IBM Tivoli Change and Configuration Management Database (CCMDB) in this issue.[8]
4. *Authoritative source for a configuration attribute*—With the presence of multiple management systems providing facets of configuration and relationship attributes of a resource, it is necessary to allow the authoritative provider to be designated for any particular attribute.
5. *Access*—The CMDB provides open interfaces to access data in the CMDB and to import data into the CMDB. Data access is enabled through several interfaces based on Java** and EJB** that allow processes and other OMPs to both access and populate the CMDB data. In addition, built-in capabilities to load data from a standard XML format (Identity Markup Language—IDML) of the CMDB data model is also provided.

Integration of service management processes with OMPs

Tasks performed as part of ISM processes leverage IBM OMPs and third-party products for task automation, thereby improving the overall efficiency of service management. Monitoring, event infrastructures, provisioning, distribution, availability, workload management, replication, backup, and security are among the pervasively deployed OMPs. For example, the deployment of a large-scale software update may utilize IBM Tivoli Provisioning Manager for Software to automatically distribute this software update to large numbers of desktops based on a schedule. The ISM architecture allows the definition of logical management operations (LMOs) that provide an interface between the service management process and the OMPs that carry out the operation. A Web-services-based SOA is used to implement these interfaces. This allows a loose coupling between the process and the OMP that provides the function, thus allowing an implementation to exploit best-of-breed OMP technology while maintaining process consistency.

To enable this loose coupling, the LMO interface is implemented by using an integration module. The integration module performs two key functions:

1. It implements one or more calls to one or more OMPs by using the native interfaces of the OMPs, which could include command-line interfaces or application-programming interfaces (APIs).
2. It maps the call arguments (provided by the process and based on the CMDB resource model) to arguments that are understood by the OMP. For example, a globally unique identifier (GUID) used by the process and CMDB to identify a server may need to be mapped to an object identifier that is used by the IBM Tivoli Provisioning Manager for Software to internally identify the same server.

The evolution and automation of processes will require additional LMOs and implementations. The ISM architecture supports the installation and configuration of these integration modules to interact with specific processes and tasks.

User interfaces

To improve the effectiveness and efficiency of processes, it is important to provide UIs that allow effective collaboration between users and enable

them to perform their tasks effectively. The ISM architecture uses a portal to consolidate the UIs where users can view and claim tasks assigned to them and from which they can launch applications and view the topology of resources and their relationships. For example, to assess the impact of an RFC, the user may need to visualize the topology of the relationship of a resource to a business application as well as other changes scheduled on the resource by other RFCs. A portal allows the user to generate these two distinct views in the same workspace, which improves the user's effectiveness in carrying out the change impact assessment activity. The standards-based portal (JSR-168)[9] allows the user to customize the UIs.

The UIs also enable user efficiency by enabling activities such as the invocation of OMPs to automate tasks performed by the user, send notifications to other users, and refer to attachments and documents created as part of the process.

Design tooling to promote ease of use for process configuration

As described earlier, it is necessary to provide tools for creating and modifying processes, activities, and tasks. The ISM architecture leverages WebSphere tooling support (WebSphere Business Modeler and WebSphere Integration Developer) in addition to extensive tooling support for handling database configurations, conditional routing between activities and tasks, and UIs.

In addition to simplifying the configuration of data, processes, and UI components, it is necessary to preserve configuration changes in a manner that supports movement from one environment to another (e.g., from test to production). The ISM architecture enables these configuration changes to be stored as metadata (XML). Migration to a new environment will import this metadata to incorporate the configuration changes without the need to manually reconfigure the product.

CONFIGURATION MANAGEMENT DATABASE

The CMDB is the repository that maintains configuration and relationship information about IT infrastructure resources. Resources stored in the CMDB are called configuration items (CIs). The CMDB provides automated application discovery, detailed views of system, software, and service topologies, and the ability to maintain authorized states of these resources and their relationships. The CMDB enables federated access to detailed resource configuration data maintained in other management systems and data sources. Open interfaces to the CMDB provide ease of integration with process, data sources, and automation technology.

A CMDB provides authoritative and reliable information on the state of the IT configuration. To be authoritative, the quality and accuracy of data in the CMDB needs to be maintained by controlling changes through change- and configuration-management processes. The CMDB supports service management tasks such as the following:

- Assess the impact of a requested change (change management)
- Assess business services impacted by an incident (incident management)
- Audit to compare authorized and actual state of CIs and their relationships (configuration management)

As shown in *Figure 4*, the goals of the CMDB are achieved by (1) discovering resources and relationships (actual state), (2) comparing the actual state against the authorized state, and (3) invoking well-defined change and configuration processes to address any discrepancies between the actual and authorized states. In addition, the CMDB maintains links to process artifacts (such as RFCs) to help determine which process changes one or more CIs and to access related documents and artifacts (for example, a change assessment document).

The CMDB is central to the ISM architecture because it provides key interfaces to the other components of the architecture to realize service management, as follows:

- Interfaces to the IT infrastructure and existing management tools for discovery, reconciliation, and federation of existing data sources
- Interfaces to the process layer to enable process tasks
- Interfaces to the UI layer to visualize CMDB data and relationships
- Interfaces to configuration tools to manage the CMDB schema and extensibility

Several of these aspects are described in the following sections.

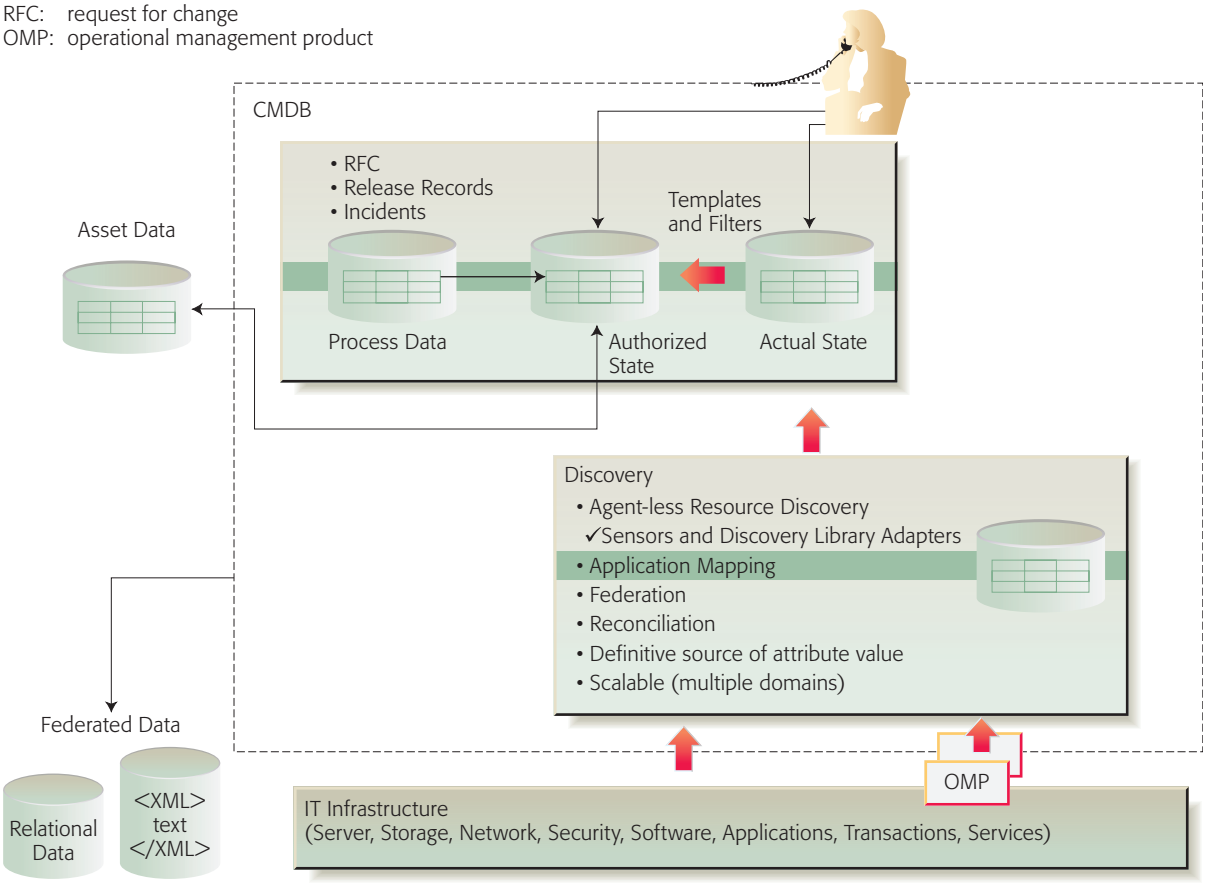

Figure 4
Configuration management database (CMDB)

Data acquisition (discovery), reconciliation, and access

The CMDB supports the following mechanisms to populate and maintain the data in these repositories:

- *Data discovery*—Information about CIs can be discovered directly from the IT infrastructure through sensors. In addition, the data import mechanisms bring in data from other sources (e.g., management tools, spreadsheets), which may already have discovered information about the CIs.
- *Data federation*—Enables logical access to CI data from another repository.
- *Data reconciliation*—Ensures that discovery of data originating in multiple sources of data (such as distinct monitoring and provisioning management tools) about the same CI results in a single system of record in the CMDB.
- *User input and application programming interfaces*—These interfaces create, read, update, and delete data in the CMDB.

An important capability of the CMDB is the rule-based reconciliation of data discovered from different sources. Multiple reconciliation rules can be specified for a resource. For example, a server may be identified by the make, model, and serial number of the server, or equivalently the MAC (media access control) address or host name. The CMDB creates aliases when multiple naming rules are applicable. This enables the CMDB to reconcile additional information sources for the same resource in the same CI.

To avoid maintaining all possible resource-configuration and relationship data in the CMDB and the resulting scalability and consistency issues, the CMDB implements data federation. Data that

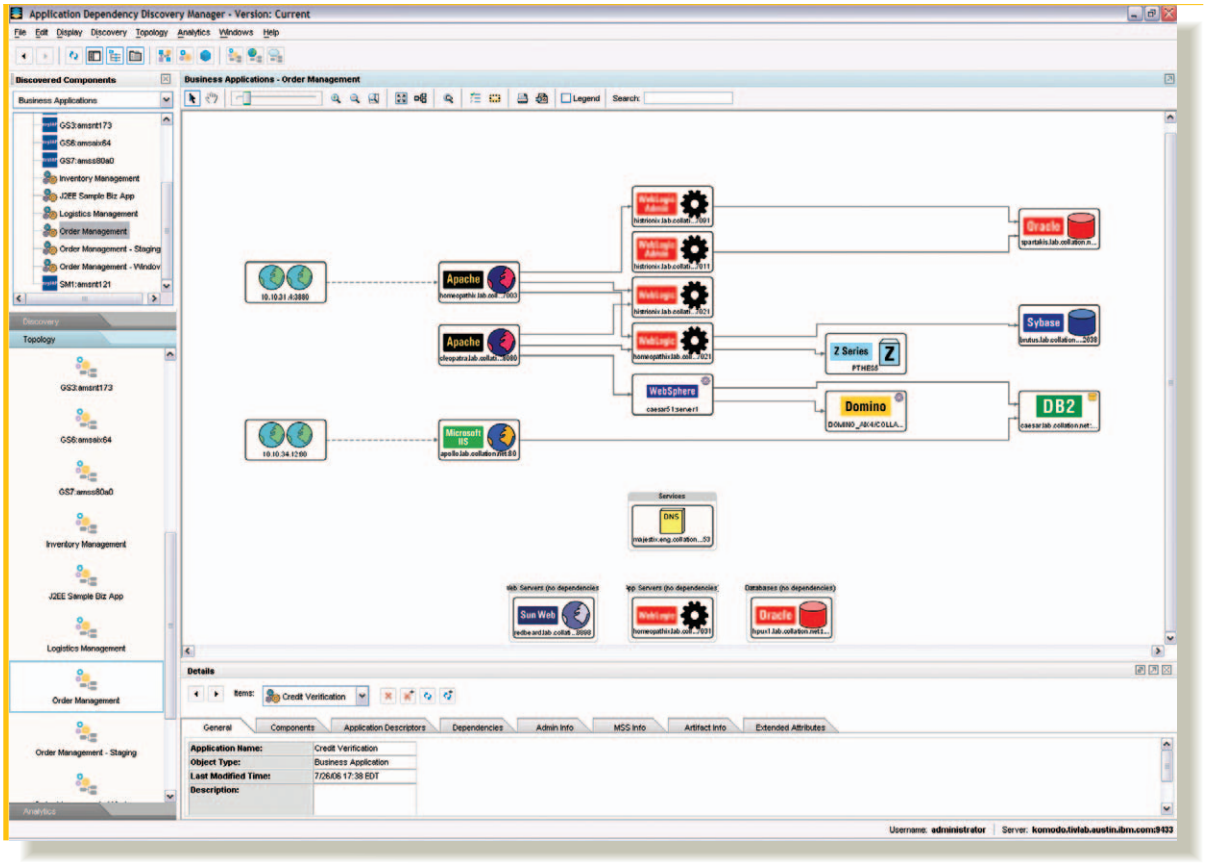

Figure 5
Visualization of application dependencies using the CMDB

changes rapidly (e.g., resource status maintained by a monitoring system) is left under the control of existing management systems; however, this data is logically made part of the CMDB (i.e., linked to the information in the CMDB). Access to this data is enabled transparently by using the federation capabilities of middleware products such as the Websphere Information Integrator.[10–14]

The CMDB also provides capabilities to maintain the authorized state (attributes and relationships) of the CI. The authorized state is always modified by change- and configuration-management processes with appropriate impact analysis and approvals. Having both authorized and actual (discovered) data about CIs and relationships enables functions such as audit and compliance checks.

Data in the CMDB is accessed by consumers, including UIs, process tasks, and OMPs, by using an object layer abstraction on the underlying database structure. The abstraction seen by the consumers is a set of Java objects and operations on these objects, including search, create, read, update, and delete. The object layer abstraction allows the consumer to be insulated from the layout of database tables and the optimizations that can be exploited at that level. The CMDB is described in additional detail in a separate paper in this issue.[8]

Visualization

Figure 5 illustrates the use of the CMDB to visualize the dependencies of a business application (an order-management application is shown in Figure 5) on the constituent IT infrastructure components (such as software applications, servers, storage, and networks). This visualization enables the user (IT staff) to iteratively drill down (i.e., view data at a greater level of detail) to specific components. For example, the user can drill down from the business service to the underlying software components and to the servers on which the software is installed. The

user can obtain details on the software and server configuration (for example the EJBs installed on a WebSphere application server).

The CMDB architecture provides unique capabilities, including the following:

- A comprehensive data model (CDM) that supports open standards and operational extensions based on field experience. This significantly accelerates a customer's implementation of a CMDB.
- Closely integrated change and configuration processes and OMPs to enable consistency of the CMDB. Additionally, the ability to support automated change and configuration processes enables the delegation of functions to management products and provides the architecture to integrate service management and autonomic behavior of resources and management systems.
- Open interfaces and standard data interchange formats.

INTEGRATION OF SERVICE MANAGEMENT PROCESSES WITH OMPS

The integration of ITIL or COBIT-like best practice processes with OMPs is a critical enabler for improving the efficiency of process activities and tasks. OMPs, such as monitoring, event management, provisioning, and license management products, allow service management processes to be applied to large-scale resource domains while minimizing repetitive labor cost and resulting errors.

The ISM platform supports three kinds of integration between OMPs and service management processes: UI integration, data integration, and functional integration.

Data integration is enabled by the CMDB. Data can be transferred from the OMP to reside in the CMDB (discovery) or can remain in the OMP but logically mapped to the CMDB (federation). This integration is described in the subsection "Data acquisition (discovery), reconciliation, and access" in the section "Configuration management database."

UI integration

Manual tasks are often performed by launching OMPs and interacting with their UIs. Examples of these include:

- While deploying a change that is requested on a server (as part of a change management process), a change deployer may want to see detailed information about this server maintained in the data center model of Tivoli Provisioning Manager.
- To identify the failing component (as part of an incident management process), a service desk analyst may need to launch into Tivoli Business Services Management.

The user interaction with the OMP UI can be optimized (i.e., redundant input and the number of panels navigated can be minimized) by launching the OMP UI in the context of one or more CIs for which the task is being performed. Launching into the OMP UI should not only display the appropriate contextual view but also pass contextual information about the CIs for which the OMP is being launched.

Launch in context is provided as a general mechanism in the ISM architecture. Launch in context can be used between service management processes and OMPs, directly between OMPs, between different processes, and between processes and the CMDB. ***Figure 6*** illustrates the failure analysis task in an incident management process to determine the business impact. The failure analysis task identifies the failing components, services impacted, and the service level agreements (SLAs) impacted. The failure analysis task UI also enables the user to launch into the OMPs (Tivoli Enterprise Portal, Tivoli Business Systems Manager, and Tivoli Service Level Advisor) to obtain additional detail about resources, their status, and the SLAs in place.

Functional integration

Functional integration enables process tasks to programmatically invoke OMPs to execute specific tasks. In conformance with an SOA approach, the invocations of OMPs are implemented by using an LMO, an abstract logical interface that is loosely coupled with the specific APIs provided by an OMP. As mentioned earlier, LMOs provide a service abstraction of the OMPs and a degree of transparency from versions, instance, and location. The architecture enables the integration module that implements an LMO (i.e., binds the LMO to a specific OMP) to be separately developed, installed, and configured within the platform. This provides the opportunity to create an ecosystem of OMP vendors to be integrated with the ISM platform. The integration module architecture includes the following key aspects:

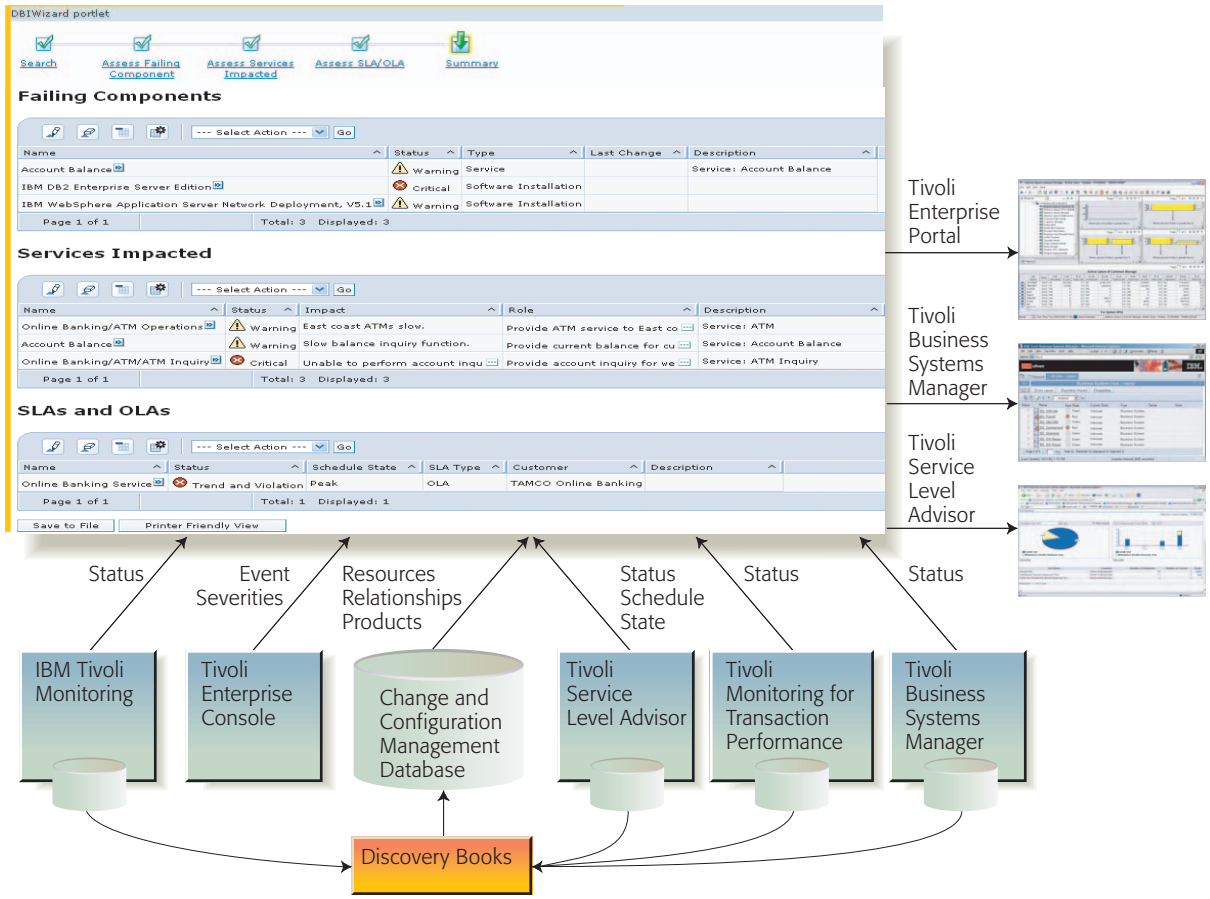

Figure 6
Launch-in-context integration with operational management products

- An integration module is defined as a Web service, using standard Web Services Description Language (WSDL) interface definitions.
- An integration module implements one or more LMOs by providing the binding and functional mapping between the syntax and semantics of the LMOs to the interfaces provided by the target OMP.
- To accomplish the binding and functional mapping, the integration module translates the process reference to a CI into corresponding identifiers for one or more resources as understood by the OMP. This is accomplished by interacting with the CMDB, which maintains the mapping between the CMDB CI identifier and the resource identifier for each OMP that can manage the resource.
- Each integration module is registered with the CMDB with appropriate information on the location of the OMP, the LMOs it supports, and the set or collection of CIs for which it supports the LMOs.

Use of an SOA to implement OMP integration is a key differentiating aspect of the ISM architecture. We expect that the increasing maturity and automation of service management will drive the definition of standardized LMOs for the various service management process domains.

SERVICE CATALOG AND SERVICE REQUEST MANAGEMENT

For most end users, their view of ISM begins with the service catalog, which lists the services offered by service providers. For an IT organization, the service catalog represents the interface with its end users. When such a user selects a service from the catalog, a service request is created and handled by the appropriate IT processes as depicted in Figure 2.

Service definition and catalog views

The ISM architecture enables services to be defined in XML with a service definition tool and stored in a service database. When services are provided by a service provider, entitlement to these services (including access to specified views of the data) can be based on criteria such as membership in an organization and customer account information. The list of services can range from simple, fully automated end-user services such as password reset, to more complex services such as provisioning or upgrade of an application environment. Service definition also includes setting up the terms of any associated SLAs, rating and billing terms, and contractual agreement templates. Note that the service definition by itself is not sufficient to bring a new service into existence. All the appropriate service fulfillment workflows and integration with internal processes and systems, as well as external service providers, have to be established before the service can be delivered.

The defined services are accessible through the service portal—where the views can be customized for the user. Users can browse the service catalog for services that they are entitled to request, view groups of services, and select a service. For example, the user could browse the catalog for user ID services, and select the password reset service. On selection, the user is asked to provide values for the attributes associated with that service. In addition to UIs, programmatic interfaces, such as add/modify/query the list of the services, are also provided.

Service catalogs can also display the cost information associated with a service. This is appropriate, for example, when the IT organization charges for the specific services rendered. In such cases, the end user can factor in the cost of the services when requesting the service.

The service catalog also provides interfaces for business users, those users who define services and create and modify the business rules associated with the service. For example, a business user can provide price information, levels of discounts available, and the kinds of resources used to satisfy a particular service request. Business users also have access to service performance data and analytics in the form of reports and dashboard views. This provides business users with metrics on how a service is performing (e.g., which services are ordered often).

Service request management

Once the user selects a service from the catalog, the service has to be fulfilled by the IT organization either through internal capabilities or by aggregating internal capabilities with externally provided (outsourced) capabilities. Whereas simple services such as password resets can be easily automated, more complex services such as server provisioning may involve requisitions, reviews, assessments, and approvals, which involve human intervention. The sequence of work involved in fulfilling the service is called the service request flow.

In addition to managing the fulfillment of a service request, service request management also provides the service requestor with periodic status reports on the progress of service fulfillment. This information may also be used to monitor the service for performance measures such as availability and utilization.

PROCESS MANAGERS

Process Managers (PMs) are applications that deliver service-management process implementations through executable workflows integrated with OMPs and the CMDB. In addition, PMs provide capabilities to track execution metrics and provide dashboards and reports that allow IT organizations to identify bottlenecks and improve organizational productivity.

A service management process is initiated by an incoming work request, An RFC is a common example of such a work request. Because IT organizations deal with a large variety of work requests, the ISM architecture provides a flexible and extensible mechanism to classify the work requests. For example, an RFC may involve hardware changes, software changes, network changes, or storage changes. Software changes can be further classified into new application deployments and application upgrades. Each type of work request requires its own set of extended attribute values, which capture all the information needed to perform the work request.

The actual fulfillment of this work request is handled by a process consisting of activities (subprocesses) and tasks. A process flow is a graph of

activities and tasks. For our purposes, a *task* is defined as a unit of work that can be scheduled and assigned. PMs provide mechanisms to allow process owners to define flow templates associated with each type of request. This gives each user organization the ability to exercise control on the way work is performed, using the organization's policies and guidelines, while still supporting the execution of end-to-end service flows. The end-to-end service *request and fulfillment flow* is realized by the initiating service request flow and the collection of flow templates in each of the PMs that are exercised as the particular request makes its way through each process domain.

The breadth and complexity of services along with domain-specific requirements makes it difficult to predefine end-to-end process workflows. Often, the activities are well-known, but the tasks vary depending upon the specific request and organization policies. The specific tasks and the sequencing and scheduling of tasks may depend on a number of dynamic considerations, including domain-specific requirements as well as personnel availability and organization responsibilities. The PM architecture provides several ways to represent the tasks performed by various people in the organization as part of the process flow. The most basic representation is in the form of work breakdown structures, which may just identify the list of tasks to be performed in an activity and the dependencies between tasks. Additional representations can define explicit workflows to support more advanced task-sequencing needs (including conditional branching).

Tasks can be assigned to either an individual or a team, typically identified by role and implemented as a Lightweight Directory Access Protocol (LDAP) security group. When the task is assigned to a team, the task appears in the in-box of all the members of that LDAP group. Users (IT staff) who participate in the process flow log into a portal UI to view the tasks in their in-boxes. The user can choose to accept or claim a task, at which point the task is removed from the work queue of other team members. When the task is claimed by the user, a UI for performing the task is launched.

Process owners and business managers can view the current status of a particular work request on the PM console and can view the tasks that have been completed and the tasks that are yet to be completed. Process execution can be monitored through the definition of key performance indicators (KPIs). They enable the identification of bottlenecks and inefficiencies and may lead to process improvements.

The CMDB provides a common repository of information that is used by all PMs. For example, the Change Management PM and the IBM Tivoli Release Management PM use information from the CMDB to determine software packages and targets that would be involved in a particular change. Additionally, the configuration management process is responsible for updating the authorized configuration data in the CMDB. This ensures the consistency of data and facilitates integration across processes. For example, when new software is deployed to a server, the CMDB is updated with the new CIs and relationships by the IBM Tivoli Configuration PM—and this information is then available to the Incident Management PM should incident tickets be opened against services or applications running on that server.

PMs support Web services interfaces for ease of integration with other processes. For example, the Change Management PM supports a Web services interface to create an RFC, to query the status of an RFC, to cancel an RFC, and so on. Similarly, the IBM Tivoli Release Management PM supports an interface to create a release, query details of a release, suspend a release, and cancel a release.

ISM IMPLEMENTATION EXPERIENCE

Our implementation of the ISM architecture encompasses almost all major components described in this paper. The implementation is built on WebSphere middleware and uses industry open standards for interfaces, the portal, data models, and process workflows. The core IBM operational management capabilities integrated include: Tivoli Monitoring, Event Management, Business Systems Management, Provisioning and Software Distribution, Storage Management, Security Management, and Network Management. The IBM Tivoli Change and Configuration Management Database (CCMDB) is an integrated offering that includes the CMDB and the change-management and configuration-management processes necessary to maintain the integrity of the CMDB. The CCMDB supports the federation or discovery of information on resources

and their relationships, the visualization of dependencies among data, and processes for managing configuration changes. PMs have been implemented for a variety of domains including release management, storage provisioning management, availability management, and capacity management.

Because the ISM implementation is loosely coupled, this allows for progressive adoption and incremental deployment. This is a common adoption pattern in various implementations. Deployments of IBM Tivoli OMPs and integration between these products has gained significant customer acceptance. The ISM implementation provides an evolutionary path for adoption of service management by integrating OMPs with the CMDB and PMs.

There has been significant adoption and realization of value from the discovery and topology visualization capabilities provided by CCMDB. This has been a critical step in the adoption of the ISM vision and architecture. The deployment of discovery and application dependency mapping provides users with the critical information required to improve the effectiveness of their IT processes. The deployment of CCMDB discovery has generated requests to expand the breadth and depth of discovery beyond the already extensive list of supported resources and management systems. Additional requirements from field implementations include the ability to control the scope of discovery and to filter the discovered data to limit the amount of information gathered as part of discovery. This has primarily been driven by scalability and the amount of time to discover large numbers of resources. Other requirements include the ability to extend the data model, to add attributes to existing resource models, and to add new types of resources. There is also significant interest in the federation and configuration management capabilities of CCMDB. Federating data sources is critical for the inclusion of customer and third-party databases in CCMDB. These deployments are pushing the need for increasingly sophisticated and automated reconciliation technology.

From a process management perspective, many customers have been looking for the flexibility to configure the tasks of a process to more closely align with their practices and policies. This has driven advanced tooling requirements to configure processes and integrate the processes with data model extensions in the CCMDB and UIs for visualization. Understanding the existing processes and interactions across organizations is an important aspect to successful process management deployment. The transformation toward best practices typically begins with implementation and incremental change to existing processes. Typically process implementations require substantial time and implementation effort because they often include organizational transformation as well as alignment with existing processes and tools and their limitations.

Our implementation and deployment experiences have confirmed the key design decisions of the ISM architecture. In particular, the use of an SOA to integrate our extensive collection of OMPs with a federation-based CMDB and process management technologies was validated. Each of the deployment requirements we have encountered is being addressed in a manner consistent with our architecture.

SUMMARY AND FUTURE WORK

We have presented an SOA-based architecture that provides a platform for aligning operations teams with industry best practices and integrates process with information and operational-management technologies. Built upon business process transformation tools, information management technologies, and operational management technologies, the IBM Service Management architecture establishes a foundation to support more advanced levels of autonomic computing. The knowledge contained in the CMDB is a critical component for autonomic management of systems. The service topology information supported by the CMDB provides a business context to establish policies to govern autonomic behavior. These policies can reflect a range of domains from quality-of-service objectives for performance and availability to security and compliance requirements. Even policies guiding the automation of IT processes can be established. Through these policies, autonomic computing extensions can be added to the CMDB, the PMs, and the operational management tools.

These policies can then be supported through closed-loop processing. The design point is to use the OMPs to enforce the policies based on the context of the business applications as defined in the

CMDB. An important aspect of this work is the integration of the autonomic capabilities with the PMs. Ideally autonomic implementations of activities and tasks provide governance and KPIs in a manner consistent with manual tasks. This approach enables the operations team to more easily monitor the actions and results of autonomic technology, thus easing the transition from a labor-intensive model toward a more automated model where IT supports the needs and priorities of business services.

Another important aspect of this work is the design of PMs with conditional branches to switch between different levels of implementations (manual, semi-automated and automated) of the same task. This also allows customers to gradually change, as organizational maturity grows, from a manual process to a more automated and even to a fully closed-loop controlled process.

In summary, IBM Service Management is leading an industry shift, causing the discipline of management systems to evolve from a technology-centric approach toward a service focus that encompasses people, processes, information, and technology. The IBM Service Management architecture focuses on simplifying the development, deployment, and management of services, reducing operational costs, and improving service levels. Technological advances in autonomic computing are an integral part of service management. The architecture and implementation will continue to evolve toward process, tools, and integration to enable self-managing goal-oriented systems.

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

**Trademark, service mark, or registered trademark of United Kingdom Office of Government Commerce, Systems Audit and Control Association, Telemanagement Forum Corporation, or Sun Microsystems, Inc., in the United States, other countries, or both.

CITED REFERENCES

1. *Foundations of IT Service Management Based on ITIL*, ITSM Library, J. Van Bon, M. Pieper, and A. van der Verrn, Editors, Van Haren Publishing B.V., Zaltbommel, The Netherlands, November 2006.
2. *Introduction to ITIL*, The Stationery Office, Office of Government Commerce, United Kingdom (2005).
3. COBIT, Information Systems Audit and Control Association (ISACA), http://www.isaca.org/Template.cfm?Section=COBIT6&Template=/TaggedPage/TaggedPageDisplay.cfm&TPLID=55&ContentID=7981.
4. Recommendation M.3050: Enhanced Telecommunications Operations Map (eTOM)—Introduction, International Telecommunication Union, http://www.itu.int/rec/T-REC-M.3050.0/en.
5. *Service-Oriented Architecture, IBM Systems Journal* **44**, No. 4 (2005).
6. IBM Tivoli Provisioning Manager: Product Overview, IBM Corporation, http://www-306.ibm.com/software/tivoli/products/prov-mgr/.
7. IBM Tivoli Unified Process, IBM Corporation, http://www.ibm.com/software/tivoli/governance/servicemanagement/itup/tool.html.
8. H. Madduri, S. S. B. Shi, R. Baker, N. Ayachitula, L. Shwartz, M. Surendra, C. Corley, M. Benantar, and S. Patel, "A Configuration Management Database Architecture in Support of IBM Service Management," *IBM Systems Journal* **46**, No. 3, 441–457 (this issue, 2007).
9. JSRs: Java Specification Requests—JSR# 168, The Java Community Process, http://www.jcp.org/en/jsr/detail?id-168.
10. A. Betawadkar-Norwood, E. Lin, and I. Ursu, "Using Data Federation Technology in IBM WebSphere Information Integrator: Data Federation Usage Examples and Performance Tuning," *developerWorks*, IBM Corporation, http://www-128.ibm.com/developerworks/db2/library/techarticle/dm-0507lin/.
11. A. Betawadkar-Norwood, E. Lin, and I. Ursu, "Using Data Federation Technology in IBM WebSphere Information Integrator: Data Federation Design and Configuration," *developerWorks*, IBM Corporation, http://www-128.ibm.com/developerworks/db2/library/techarticle/dm-0506lin/.
12. L. Haas and E. Lin, "IBM Federated Database Technology," *developerWorks*, IBM Corporation, http://www-128.ibm.com/developerworks/db2/library/techarticle/0203haas/0203haas.html.
13. L. M. Haas, E. T. Lin, and M. A. Roth, "Data Integration through Database Federation" *IBM Systems Journal* **41**, No. 4, 578–596, 2002.
14. "Data Federation with IBM DB2 Information Integrator," IBM Redbook SG24-7052, IBM Corporation, http://www.redbooks.ibm.com/abstracts/sg247052.html.

Accepted for publication March 25, 2007.
Published online July 11, 2007.

David Lindquist
IBM Software Group, Tivoli, 3901 S Miami Blvd, Durham NC 27703-9135 (lindqui@us.ibm.com).

Mr. Lindquist, an IBM Fellow, is IBM Tivoli's Chief Architect, responsible for the architecture of IT management and service management solutions and technology. Prior to joining IBM Tivoli in 2002, he led the WebSphere Edge of Network strategy and architecture in the Application and Integration Middleware division. Dave began his career with IBM in the Server Group, specializing in large-systems architecture, performance, and database systems. In 1990 he joined the IBM Software Group, where he focused on Web infrastructure, content delivery networks, mobile and wireless technology, and Internet products. His research has led to 48 patents, recognition as an IBM Master Inventor, and election into the IBM Academy of Technology.

Hari Madduri
IBM Software Group, Tivoli, 11501 Burnet Rd, Austin TX 78758-3400 (madduri@us.ibm.com). Dr. Madduri started his career as a S/370™ assembler programmer/analyst, and obtained a Ph.D. degree in 1985 from the University of Wisconsin-Madison. Since joining IBM in 1990, he played various lead technical and management roles in object-oriented systems (DSOM), data mining (chief architect of data mining products), e-commerce hubs, electronic data interchange, and IBM Global Services service development (e.g., UMI). In IBM Tivoli, he contributed to early ITIL process prototypes, which led to the current ITSM strategy. He is currently lead architect for the CCMDB product. Dr. Madduri taught undergraduate and graduate classes in programming languages, compilers, and operating systems at University of Wisconsin-Madison, St. Thomas University (Minneapolis), and University of Hyderabad (India). He has published over 20 papers and authored 20 United States patents.

Chakalamattam Jos (C. J.) Paul
IBM Software Group, Tivoli, 11501 Burnet Road, Austin TX 78758-3400 (cjpaul@us.ibm.com). Dr. Paul is lead architect for Process Managers in IBM Tivoli and guides the architecture and design of the family of Process Managers that are a part of the IBM Service Management portfolio. His focus areas include process automation, systems management, service management, compliance, and governance. Prior to this assignment, he worked on the IBM on demand automation strategy and architecture, the autonomic computing initiative, Tivoli core technologies, and the WorkSpace on Demand product line. He is a member of the IBM Autonomic and Tivoli Architecture Board, the IEEE, and the ACM and holds more than a dozen patents. Dr. Paul joined IBM in 1993, initially working on microkernel operating systems. He has a Ph.D. degree in computer engineering from Carnegie Mellon University in Pittsburgh, Pennsylvania and a B.Tech degree from the Indian Institute of Technology in Chennai.

Bala Rajaraman
IBM Software Group, Tivoli, 3901 S Miami Blvd, Durham NC 27703-9135 (balar@us.ibm.com). Dr. Rajaraman has been with IBM since 1992 and is currently a Distinguished Engineer responsible for the architecture and design of Enterprise Systems Management solutions. His focus areas include IT service management and provisioning and automation solutions. In the past he was involved in the performance aspects of the System z™ and WebSphere. His areas of interest include communications technologies, systems performance, autonomic computing, systems management, and on demand computing. He has a Ph.D. in computer engineering from Clemson University. ■