The process of building a Process Manager: Architecture and design patterns

C. J. Paul

Process Managers are applications that implement and manage process flows integrating people, information, and technology. They allow customers to improve organizational productivity while satisfying governance and compliance requirements. This paper describes the design challenges in building Process Managers in contrast with workflow-based applications. A methodology is presented which includes using industry best practice processes, usage scenario analysis, and architecture and design patterns that have been derived from lessons learned over the last few years of building Process Managers. Although the approach and design methodology are described for the information-technology-enabled service management domain, the overall design principles are expected to be of interest for most process management domains.

INTRODUCTION AND BACKGROUND

Over the years, companies and organizations have implemented processes as a way to coordinate and sequence the work done by a collection of people and organizational entities. Initially, all of these processes were manual, but over time, components of the process have been automated by using tools. In domains where throughput and efficiency are key considerations, much work has been done in analyzing and optimizing these processes.[1] Common examples of this work include the order management process for an online store[2] or a loan origination process for a financial institution.[3]

Providing unique business services organized in a unique way, each company has tended to define custom business processes from scratch and to build custom applications to help them implement and automate these processes. These applications are known as *process-based applications*. Sometimes, process-based applications are built using workflow technologies (e.g., Web Services Business Process Execution Language [WS-BPEL[4]]), and execute using workflow engines provided by products like the WebSphere* Process Server.[5] The subset of process-based applications built using workflow technologies is referred to as *workflow-based applications*. There is a substantial amount of literature that describes how to build workflow-based appli-

cations,[6] including modeling,[7] building, deploying, and monitoring them.[8]

In recent years, companies in most industries (including the retail, finance, health-care, and manufacturing industries) have become dependent on information technology (IT) to provide fundamental business services to their customers and end users. As the underlying IT infrastructure has increased in complexity, it has becoming increasingly critical to effectively and efficiently coordinate the work done by the different parts of the service management organization (e.g., the server team, network team, storage team, and application teams) in building, deploying, and operating the IT-enabled business services. Without proper processes and coordination, expensive service outages can occur due to erroneous changes made to the IT infrastructure supporting those business services. To address this problem by using an approach similar to business processes, companies have tended to define custom service management processes and build custom service management applications (e.g., an in-house change management application) to help them automate parts of these service management processes. Custom processes and applications such as these are expensive to build and maintain.

This trend is gradually shifting. Due to increasing costs and operational pressures, companies are investigating ways to implement processes using commercial "off the shelf" tools to address their internal needs. In recent years, there has been increasing interest in understanding and leveraging industry best practices like the IT Infrastructure Library** (ITIL**),[9,10] Control Objectives for Information and Related Technology (COBIT**),[11] and enhanced Telecom Operations Map (eTOM**)[12] to improve and standardize internal service management processes. Recent laws requiring compliance with certain legal requirements[13] are also contributing to increased interest in using process-based applications, as they can be used to demonstrate that the appropriate compliance controls are in place.

For this purpose, companies have been turning to vendors who have built process-based applications in order to quickly implement their service management processes. Several vendors provide process-based applications for service management, including Hewlett-Packard Development Company, L. P.[14] and BMC Software, Inc.[15] Exploiting experience gained from hundreds of implementations of service management solutions, the IBM service management strategy[16] takes this concept a step further and provides a set of process management products known as *Process Managers* (or PMs), which represent a particular vendor instantiation of a general class of process management applications.

Process Manager concepts

Process Managers (PMs) are applications that provide flexible "out of the box" implementations of best-practice processes to help customers integrate and automate processes more quickly than building them from scratch themselves. Reference 17 provides the overall architectural context for IBM Service Management and describes how PMs relate to other components of the IBM Service Management architecture.

Within a PM in the IBM Service Management architecture, executable process flows (using workflow technologies) are used to sequence through the activities and tasks in the process and to provide the right information to the right user at the right point in the process. If a process task is a manual task (including those with automation assistance), the PM provides a user interface to allow users to perform the task and complete the process step. Information is aggregated from different sources to provide users with the contextual information that can help them complete the process task quickly. Tool integration modules are used to interact with the appropriate external tools to extract information for analysis and display in order to take action on the environment (e.g., to perform a software distribution task). PMs leverage information about configuration items (CIs) in the configuration management database (CMDB) and update information in the CMDB after authorized changes are made.

Several PMs have been built and shipped by IBM over the last few years—for example, the IBM Tivoli Release Process Manager[18] and the IBM Tivoli Storage Process Manager.[19]

Related work

Process-based applications and process management applications have been built for domains such as human resource management, supply chain management,[20] and customer relationship manage-

ment.[21] Such applications also exist in the domain of IT management, notably for incident management, problem management, and change management,[14,15] with generic request management, routing, and approvals being the predominant focus.

Process management applications are different from custom workflow-based applications and from workflow runtimes, such as various industry WS-BPEL engines[22] and the IBM WebSphere Process Server. Custom workflow-based applications can focus on a particular process being implemented and optimize the implementation for that specific case. At the other end of the spectrum, workflow runtimes provide the ability to run many different process templates and provide infrastructure services, such as logging, exception handling, and escalation management, but they have no semantic understanding of the processes themselves.

Whereas custom process-based applications are often built for a particular process and a particular customer, PMs need to be easily adaptable to multiple customers. PMs also have to deal with the complexities and variations of the IT environment, which necessitate a high degree of variability in the detailed task sequences executed for different requests within a single process domain. Process automation in the IT environment also presents another set of unique challenges, given the large number of technologies and tools that have to be integrated into a PM. To deal with this, not only do PMs have to be adaptable, flexible, modular, and extensible, but also each PM has to effectively support a family of process flows and, at the same time, balance compliance requirements with the need for local autonomy for the supporting service management organizations (such as the server, network, storage, and application teams). Thus, PMs need both to understand the semantics of a particular process domain and to perform process management functions, such as managing flow templates and flow variations based on runtime context.

In the process of building PMs, we observed certain design patterns that provide for critical functional capabilities. These capabilities include leveraging best practices, adapting to the challenges of process management in the IT environment, and providing commonality, integration, and consistency of behavior throughout the PMs. This paper discusses the lessons learned and the architecture and design patterns that have been derived from our experience in building PMs and deploying them in customer environments.

PROCESS MANAGEMENT LIFE CYCLE

To manage their processes, organizations need to start by understanding and documenting them. They may focus on either the current set of processes (the as-is view), or the desired set of processes (the to-be view). Hence, the first step is often capturing, editing, and publishing the processes of the organization in a standard form by using some form of document repository to hold the process documentation. IBM provides tools like the Rational* Method Composer[23] and the WebSphere Business Modeler[7] to help customers capture and publish their processes. The WebSphere Business Modeler also provides modeling, simulation, and analysis capabilities that can be used by process consultants and business analysts to study and optimize a customer's business processes.

Starting from known best practices

In determining the desired set of processes (the to-be view), one can either start from scratch and model and analyze the relevant service management process or take advantage of published best practices. In the IT-enabled service management domain, a set of industry best practices has been documented in the form of the ITIL Library.[9] Similar best practices also exist for the telecommunications industry as described by eTOM,[12] and for governance, compliance, and process improvement as described by ISO20000,[24] COBIT,[11] and Capability Maturity Model Integration (CMMI).[25] IBM completed a thorough review of these externally published best practices, reconciled differences between them, identified roles, work products, inputs, and outputs, and documented these in a reference model called the Process Reference Model for IT (PRM-IT). The PRM-IT content is available through a tool called the IBM Tivoli* Unified Process (ITUP[26]), and content for it can be authored with a tool called the IBM Tivoli Unified Process Composer (ITUPC), as shown in the left part of ***Figure 1***.

For example, ITUP and ITUPC describe the top-level activities for change management (e.g., accept and categorize change, assess change, approve and schedule change). Each of these top-level activities is further decomposed into task-level workflows,

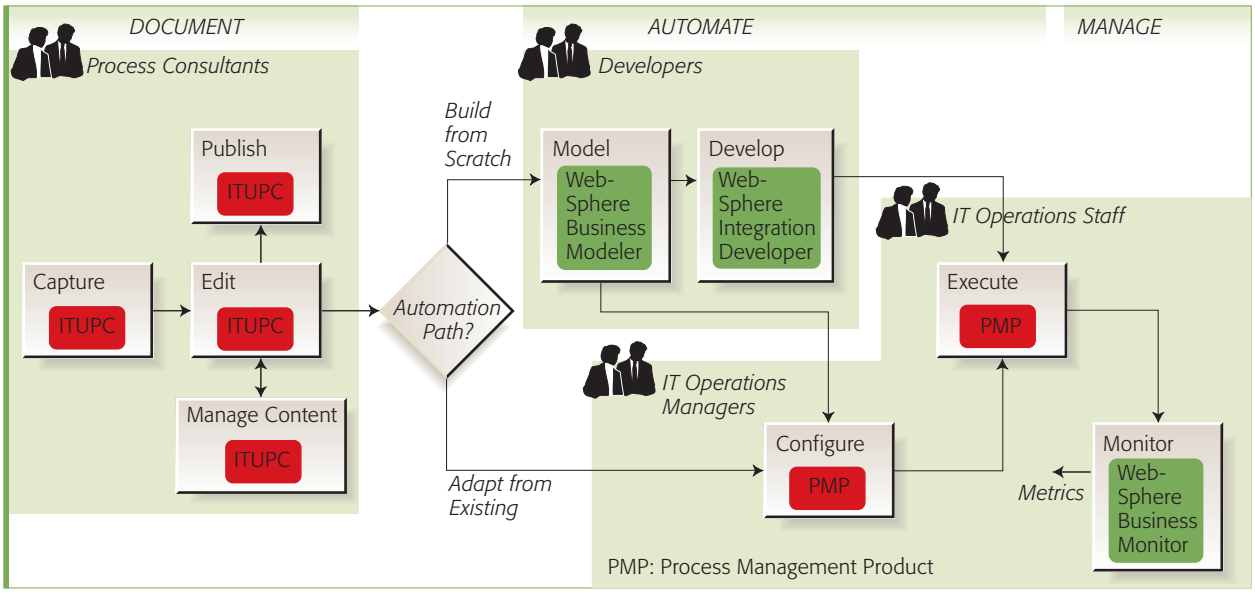


Figure 1
Process-management life cycle

performed by users in identified roles, such as change manager, change analyst, change implementor, and change process owner. The ITUP also describes the inputs and outputs and the key work products and tools needed to perform a task or activity.

Scenarios are described to show how different processes and tools work together to allow customers to accomplish an end-to-end objective. For example, an incident is recorded at the service desk and analyzed, a diagnosis of the problem is made, a change request is opened to create the solution, the solution is developed, and finally, the solution is distributed with the appropriate systems management tools.

With the wealth of content in the ITUP, process consultants can start with a well-documented best practice process and adapt it to a particular customer's needs, instead of having to define these IT-enabled service management processes from scratch.

Role of Process Managers

Once a best practice process is understood and selected, organizations then have to determine the best way of implementing that process. For organizations that need to automate their processes, it is often necessary to hire developers to build workflow-based applications from scratch. This requires organizations to go through a full development cycle to model, simulate, develop, deploy, and execute these processes, as shown in the top half of Figure 1. This is often an expensive proposition, requiring advanced development tools, such as the WebSphere Business Modeler[7] and the WebSphere Integration Developer[27] to build workflow-based applications.

An alternate option is to use a prebuilt process management product (namely, a PM) that provides an implementation of a particular process of interest, along with significant process flow management and automation capabilities. Such products need to provide IT operations managers with the ability to reconfigure process flows as needed directly through the PM GUI (graphical user interface) without going through an expensive development cycle. Reconfigured processes need to remain consistent with corporate guidelines to ensure compliance with corporate and legal requirements.

Once the PM is installed and configured, it is ready for use by the IT operations staff to perform process tasks in their daily operations. The PM is responsible for routing tasks to the right user and keeping track of the progress of the tasks assigned to different users participating in the process. Process

- Challenge #1: Process Variability and Consumability
  - Best practice implementations vary from customer to customer.
  - Fixed process flows are not practical.
  - Processes must be easily configurable and modifiable without a code development cycle.
- Challenge #2: Bridging from Abstract Process to Day-to-Day Operations
  - In the real world, workflows vary by request group.
  - It is not feasible to model all flow variants a priori (especially for diagnosis activities).
  - Modularity and reuse of process tasks and process flow segments is important.

Figure 2
Challenges in building Process Managers

execution metrics are often gathered for analysis and reporting, both by the PM itself and by external tools like the WebSphere Business Monitor, as shown in the right half of Figure 1. Analysis of these metrics is used to understand process bottlenecks and can be used to re-engineer processes to improve organizational efficiency.

Early results in translating best practices to implementation flows

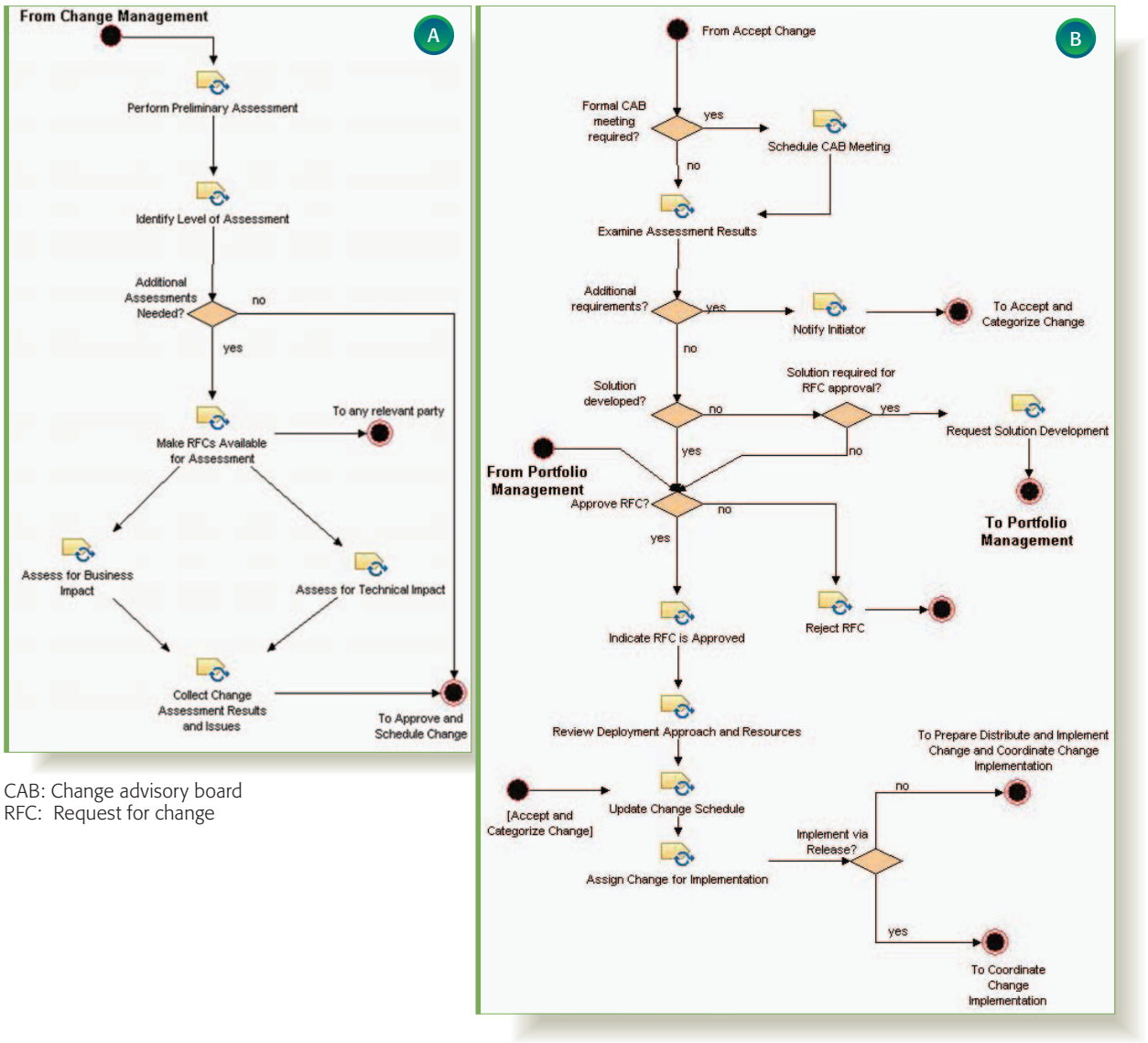
In building the first wave of PMs, we learned some lessons that caused us to go back and rework their design before the product could be shipped. Some of the challenges involved are highlighted in *Figure 2*.

Our initial iterations were literal codifications of the best practices processes into fixed workflows. We started by building a workflow-based application that reflected ITIL and PRM-IT best practices (similar to the work of many other vendors). However, because customers each have their own view of the best practices for their organization, it is not practical to have a fixed workflow implemented in the application. The workflow would have to be modified before it could be used within any particular customer's environment. This presents a set of requirements for building a general-purpose PM product which is often not encountered when building a custom workflow-based application to solve a particular customer's problem.

We also found a "disconnect" between the high-level reference process documented by organizations and the daily activities and tasks performed by the IT staff. This disconnect comes about because the reference process is an abstraction of the daily work performed, acceptable from a documentation perspective, but not for building a commercial product to help organizations implement and automate the process at a detailed level.

To understand this disconnect, we spent time analyzing the daily work of the staff. We evaluated the detailed operational processes from the hosting centers run by IBM Global Services (IGS), as well as detailed operational processes from several of our large customers. In one instance, we found that IGS had to go through 122 process steps to provision a new server for the hosting center; this involved a great deal of responsibility passing between different teams. For one of our large customers, it could take up to 43 days to provision a new server. Responsibility was passed among 21 teams, and often several times between these teams. For example, for change impact assessment, detailed assessments had to be collected from the server team, the storage team, the network team, and so forth, before the final assessment could be completed. Similarly, the workflow for deployment included process steps performed by the server hardware teams, the server operating-system team, the networking team, the middleware teams, and the applications teams.

In a series of analysis exercises, we compared this information with the available codification of best practices. One of these analysis exercises involved a few detailed workflows for change management; specifically, the workflows for the *assess change* and *approve and schedule change* activities. The reference process workflows for these two activities are shown in *Figure 3* to illustrate their differences, the level of refinement needed, and the level of reuse possible.



CAB: Change advisory board
RFC: Request for change

Figure 3
Change-management workflows from ITUP:
(A) *assess change flow* and (B) *approve and schedule change flow*

One of the observations we can make in inspecting these reference workflows is that in the case of *assess change*, the reference workflow is at a high level of abstraction and has to be refined to reflect daily operations. For example, the activities *assess technical impact* and *assess business impact* need to be refined further into unique detailed workflows for the particular kind of change request being processed. In contrast, when we review the workflow for *approve and schedule change*, we find that most of the tasks are generic enough that they can all be performed in the same way for all of the different kinds of change requests that we are expecting.

Our early prototyping and analysis led to the conclusion that PMs need to address the following key requirements:

- *Process refinement*—Before a reference process is converted into a workflow, it has to be refined further to a level of detail that represents the daily work of the organization.

- *Request-based variability*—We found variability in the tasks and people involved, based on the particular type of request being processed and, sometimes, even on the attributes of the particular request. Although ITIL was able to capture the variability of process flows for changes characterized as major, minor, or urgent, we found that this categorization was necessary, but not sufficient. Thus, request classification and typing are necessary, and each request may have its own detailed operational process flow (also referred to as a *runbook flow* or an *operational process variant*).
- *Request-specific automation tasks*—We found that the particular automated tasks used in the process flow varied, based on the type of request that was being handled. For example, a change request for additional storage would be integrated with storage management applications, while a change request for operating-system patching would be integrated with a patch deployment application.
- *Process-flow modularity and reuse*—We found that it is necessary to be able to modularize process flows for different activities and to structure activity-level flows to allow for reuse in different process flows for different request types.
- *Flow configurability*—Domain specializations are inherent in IT management. It is necessary to provide the local operations managers and technical leads with the ability to reconfigure and modify the process flow based on the needs of a particular request type. For example, the process flow for deploying a new operating system is different from the process flow for upgrading a complex application.
- *Balancing configurability with compliance*—The PMs must be able to report on compliance with the Sarbanes-Oxley Act, even if the process flows are modified. To support such compliance scenarios, corporate process owners should have the ability to "lock down" particular parts of the process or flag certain approvals as being necessary.
- *Role families*—We found that role definitions in a reference process actually represent role families. For example, the role of change manager is fulfilled by a large number of people spread across multiple organizations. For business reasons, we found that the change manager role could be segregated by business units; for example, the change manager for the credit management business unit was segregated from the change manager for the branch-banking business unit. In other cases, we found role segregation based on technical competencies; for example, the change manager for networks, the change manager for storage, and the change manager for applications were each segregated.

Beyond the particular requirements which resulted from the preceding analysis, several requirements came from numerous customer engagements. New technologies and initiatives like autonomic computing and products under development for emerging service-catalog offerings were also key sources of additional requirements for PMs.

Gradual automation support was required. The PM can be integrated with operational management products at different levels. Deep functional-level integration is the most sophisticated type of integration, and is also the most expensive to enable. Since deep functional integration with all products cannot be achieved instantaneously, it is necessary to come up with an approach that supports gradual automation.

Even if automated functionality is present, it is not foolproof. Many factors, having varied visibility, may affect the automated action. Hence, users do not always trust full automation, especially in complex environments, and need a way to initially perform tasks with the assistance of automation. They can then gradually delegate responsibility to the system to perform the task without human intervention as trust is earned.

When customers start providing services to their end customers (and internal users) through a service catalog, they expect to use their existing operational processes to fulfill various requests from the catalog. Service request flows thus invariably need to traverse multiple PMs to fulfill a particular service. Therefore, it was necessary that the architecture and design support seamless integration with service catalogs.

The preceding requirements were the key motivators for the PM architecture and design patterns that were developed.

OVERVIEW OF ARCHITECTURE AND DESIGN PATTERNS

We created a detailed PM architecture to address customer and technology requirements. This section

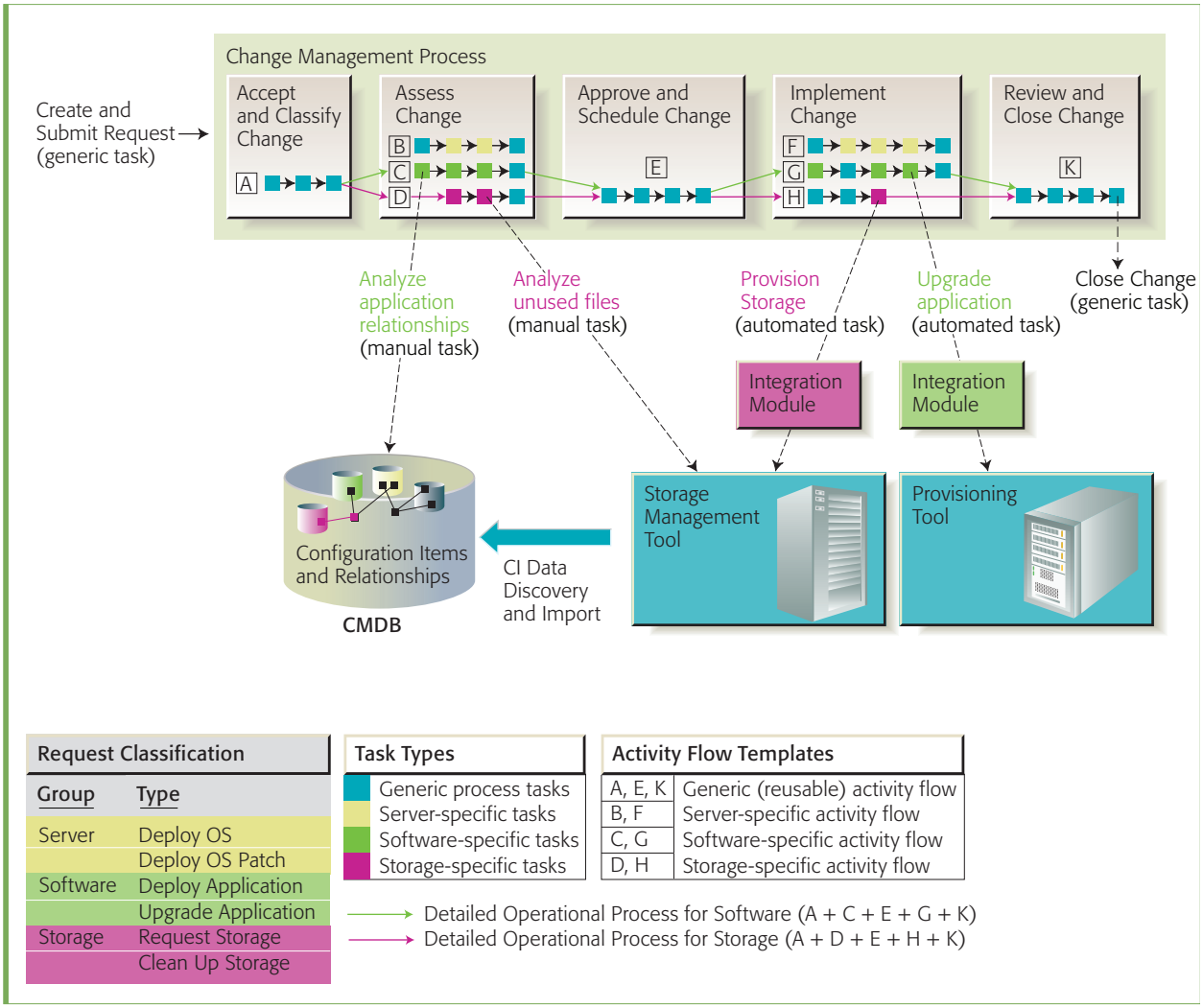


Figure 4
Process Manager structure and concepts

provides a short overview of that architecture. In a subsequent section, we describe how this architecture can be applied to build a PM for a new domain (for example, release management).

To build a PM, it is necessary to understand the types of requests handled by the organization and the particular sequence of tasks and activities that need to be performed in support of each request. In a well-structured process implementation, that detailed sequence follows the same top-level structure as the reference process and represents a refinement of the reference activities and tasks in support of that particular request type. This is a key design principle to enable flow modularity and reuse.

We use the term *process flow variants* to describe these permutations of tasks and process flows for different request types. A top-level process may have a number of detailed operational process variants that reflect the daily tasks performed for different types of requests. These flow variants are represented as flow templates in a PM.

Figure 4 shows the approach for structuring PM flow templates. The requests are classified into different groups and types, and for particular activities where type-level variability is important, there are multiple flow templates to perform that activity. For example, the *assess change* activity has three flow templates (B, C, D) because the specific tasks to assess the change request differ based on

the type of change. Software upgrade requests require tasks to analyze the application relationships in the CMDB to determine the overall business impact. Storage requests require (tool-assisted) tasks to analyze unused files, done with the help of a storage management tool. Tool-assisted tasks can be implemented either through custom task types with unique graphical user interfaces or as a launch-in-context to the supporting application. Later in the process, for the *implement change* activity, there are again three flow templates (F, G, H). Each flow template contains automated tasks that interface with systems management tools via integration modules. The automated tasks in each flow template are unique to the request type. For example, storage provisioning tasks are handled by a storage management tool such as TotalStorage* Productivity Center, whereas application upgrade tasks are handled by a software provisioning tool such as Tivoli Provisioning Manager. The figure shows that in cases where the generic flow is adequate (e.g., based on our analysis of the reference process flow for *approve and schedule change*), it should be possible to reuse that portion of the flow template (specifically that activity-flow template) for all of the request types.

Thus, an end-to-end process-flow template becomes a permutation of individual activity-flow templates, in which some activity-flow templates may be generic and others may be specific to a particular request type.

The refinement of a reference process into a detailed operational process is all the more necessary as one considers automation in the process flow. Approaching the problem in a top-down manner, we initially define the process as a fully manual process to identify the major activities and tasks to be performed by humans. As such, it is acceptable for some of the activities and tasks not to be fully specified and for the semantics to be somewhat loosely defined. When automation is considered, it becomes necessary to fully specify the detailed attributes and semantics of the task for automation; otherwise, any attempt to automate may not work or may have unintended side effects.

There are two key aspects to consider in the automation of processes. One is the making of the functional call itself; the other is having the data to make the call. On the functional side, the IBM Service Management architecture[16] uses the construct of integration modules to perform actions on the infrastructure resources by using various systems management products. The interface to the integration module from the process layer is called the logical management operation (LMO).

The functional calls to integration modules contain references to *configuration items* (CIs), an ITIL-defined term for describing anything in the infrastructure to be managed, which includes a broad spectrum of resources, including servers, networks, storage, software, and logical business applications. Information about CIs is stored in the CMDB. Some of this information ("discovered CIs") may be discovered from the environment by using discovery adapters for resources or adapters that connect to other management tools like Tivoli Provisioning Manager. In this case, discovered information for a set of resources can be imported directly from the management tool.

PMs need to be inherently extensible. It is not practical to build a PM that is able to handle only a particular set of scenarios and use cases for automation. It should be possible to extend the PM to handle additional scenarios without needing a drastic revision. Resources like storage not only need to be managed for changes, but also require monitoring, incident handling, problem identification, and corrective action. Thus, from this perspective, extensions to each of the PMs are necessary to support resources like storage. A new resource type can be handled in this architecture by creating a new set of request types, a new set of CIs, discovery adapters, task types, flow templates, and integration modules.

THE PROCESS OF BUILDING THE PROCESS MANAGER

In this section, we address the overall methodology of building a PM, starting with scenario analysis.

Business scenario and use case analysis

We describe the process of building a PM for a new process discipline, for example, release management. One can view the release-management process from a best practices perspective as described by ITIL and ITUP. The main activities in release management are *plan release, design and build release, accept release, plan release rollout, commu-*

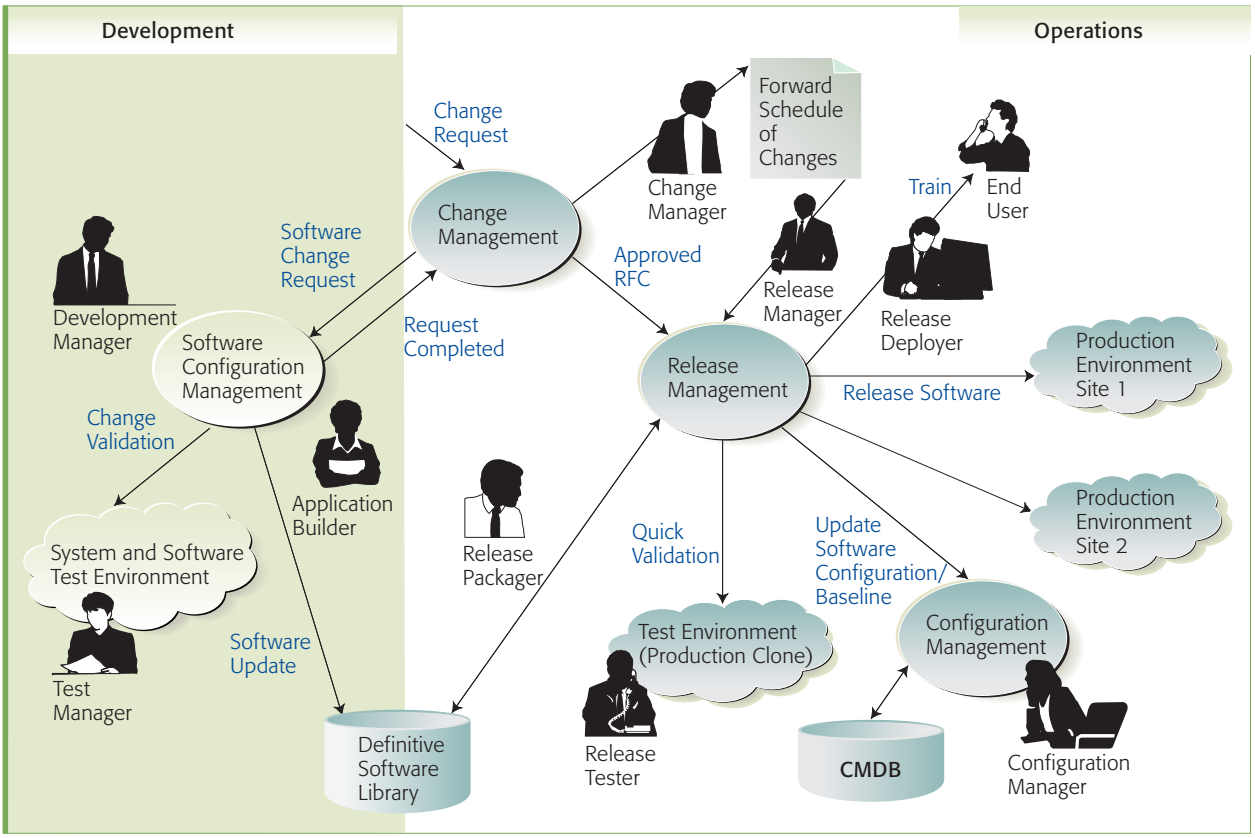


Figure 5
Release-management problem outline

nicate, prepare, and train for release, and *distribute and install release*.

It is necessary to consider the organizational, infrastructural, and tool-related context in which this process is being applied; for example, the kinds of releases to be managed by the organization. Are they hardware releases, software releases, or both? For software releases, are they new application deployments, patches for existing applications, or upgrades to existing applications? The different kinds of releases that an organization is undertaking yield information which is used to create the request classification (request groups and request types).

We focus on the particular case of the organization trying to deploy new applications. We need to understand the operational scenarios for new application deployment. Are these applications developed in-house, or acquired from third-party vendors? How will the application be packaged for deployment? What distribution tools is the organization planning to use? To how many sites does this application need to be deployed? What are the roles of the people involved? What tasks do they need to perform?

The release management problem is captured in ***Figure 5***. A change request to upgrade an application results in a software change request being created in a configuration management system (for example, such as those provided by Rational ClearQuest* and Rational ClearCase*). Once the internal application development is complete, it is handed over to the operations team, where the release packager takes the application binaries, determines the deployment configuration, and creates the deployment packages. These are validated in a test environment and later put into production at multiple sites. Once the deployments are completed, The CMDB is updated to reflect the new version of the application.

To support this scenario, a release process manager must support the categorization of the types of

application deployment requests, allow for specifying the particular set of user roles and access privileges, capture the sequences of tasks that need to be performed in the form of detailed flow templates, and structure the flow templates within the context of the top-level release-management process. In this scenario, all the specific tasks to be performed are within the context of the activities in the process.

The PM must also allow for specifying the particular software packages and targets as CIs, support the construct of a DSL (definitive software library), and support integration to deployment tools like Tivoli Provisioning Manager or Tivoli Configuration Manager. User interfaces must be provided to interact with each step in the process flow and to enable users to perform tasks with the right tools at the right point in the process. The ability to create a deployment plan across multiple sites must also be supported. Such a deployment plan should identify the site, distribution unit, person assigned, tool instance, and schedule window.

Scenario and use-case analysis helps us determine the request types, process artifacts (e.g., DSL, rollout plan, distribution units), CI types (e.g., software, servers, or network), process flows, user roles, and the particular kinds of user interfaces that are required to be part of a PM which supports a particular scenario. If the scenario were different, for example, if we were dealing with releases that involved servers, networks, and storage, then the particular tools with which the PM would have to be integrated would be different, the flows would be different, the particular task user interfaces would be different, and so on. Thus, detailed scenario analysis and use-case analysis are used to determine the right set of PM components to build. It is this combination of elements that gives the PM the richness and power to provide end-to-end process management and automation support with CMDB integration.

Process artifacts, CIs, and relationships

Once the scenario analysis is complete, the kinds of process artifacts, CIs, and relationships that need to be built in the PM should be fairly clear. The next level of detailed design represents these process artifacts and relationships and the required CIs in the CMDB. An ITIL-style CMDB should allow for the representation of both authorized CIs and discovered CIs. A comparison of the two may be done when configuration audits are performed.

Each PM has its own set of process artifacts that it needs to create and manage. Examples of process artifacts (for release management) include release requests, release details, release approvals, the master release calendar, and distribution details.

In addition, the PM also needs to create and maintain the relationships between the process artifacts and the CIs. For example, when an RFC (request for change) is opened for a particular CI (e.g., server x), an RFC object has to be created in the database, and a relationship object has to be created describing the relationship between the RFC object and the target server x.

The information embodied in these objects can be used by the PM to display information about the set of RFCs open for a particular server, to view the history of changes made for a particular server, and so on.

PMs are also responsible for creating and maintaining relationships between process artifacts. For example, an RFC may be created to respond to an open incident involving a particular application. In this case, a relationship has to be created and maintained describing the relationship between the incident object and the RFC object.

Request management

Once the main process artifacts and relationships have been designed, the next step is to design the requests that initiate the process flow. All PMs deal with requests of some kind. This is because PMs represent and manage the work done by an organization. Depending on the process discipline, there may be a large number of request types. To handle this large number, it is useful to create a classification hierarchy to classify the different types of requests. For example, in change management, requests can be grouped based on resource competencies, such as server, network, storage, and applications.

Within the application group, there could be additional types of change requests, for example, deploying a new application (e.g., a simple J2EE** application for single-machine configurations or a complex J2EE application for multimachine

configurations, a native Microsoft Windows** application, or a native Unix** application), upgrading an existing application, deploying middleware (with or without hardware prerequisites), deploying application content (e.g., HTML [Hypertext Markup Language], Word templates, Excel** files, etc.), or deploying patches (operating-system patches, middleware patches, or application patches).

In reviewing the request classification, we note that the different types of requests have different data needs. For example, the information that needs to be collected from the requestor for deploying a new application is different from the information to be collected for upgrading an application. Thus, each request type can have associated with it a set of unique request-specific attributes (beyond the generic attributes, such as the submitter name and date, which are collected for every request type).

For customers who might have a service catalog deployed, the particular request types within a PM should correspond to particular service types in a service catalog and should represent the portion of the work to be done within a process domain in support of that end-to-end service flow. Reference 28 provides some background and description on how service flows relate to process flows.

Depending on the implementation, request management may be handled either by a single (common) request management application for all process managers or by a custom request management application for each process discipline. The implementation path chosen depends on the level of sophistication in specifying request types and extended attributes that is provided by the different process management applications.

Expression and management of work in a process

Once a request has been specified, the next step in PM design is to associate that request with a particular way of fulfilling it. We first consider the case in which the fulfillment is performed by executing a workflow. IBM Process Managers support the concept of *process flow templates*. These templates specify the sequence of activities and tasks that need to be performed to fulfill the intent of the particular request.

Given a collection of flow templates, the matching of a request type to a flow template could itself have a level of sophistication associated with it. A common (default) design pattern associates each request type with a default process flow template. In advanced cases, the flow template could be automatically selected based on a set of rules that operates on the attributes of the incoming request. The request classification may simply be one of the attributes. Other considerations may include the urgency of the request, the type of the requestor, the organization to which the requestor belongs, or the CIs that are affected by a request.

To support autonomic-computing scenarios, there could be multiple flow templates defined for each request type, with each flow template providing different levels of automation, ranging from fully manual flows to fully automated flows. A change-management process flow for simple, standard changes is an example of a fully automated flow. Such changes are considered preauthorized and can be processed automatically. In such cases, the flow represents all the major activities of the change management process, with each of the steps automatically executed and logged.

Representing processes in the form of workflows is just one form of expression. In many cases, while it may be possible to describe an activity-level process flow at a high level, it may not be possible to codify detailed task-level workflows beforehand. For example, when an incident is diagnosed, the exact flow of tasks within an activity is often not known ahead of time, but will need to be determined dynamically based on the incident diagnosis work performed by an incident analyst. A number of incident analysts who are subject matter experts in different areas may need to pass the work to different members of the team based on the investigation results of an earlier step. In such cases, a detailed formal workflow that specifies every task and its sequence cannot be specified beforehand, and other forms of work assignment and expression may be more appropriate. Such a scenario, where the task sequence for the top-level incident management process is determined in an ad hoc way, is called an *ad hoc flow*.

In other cases, the particular set of tasks to be performed may be known in advance, but not

enough information may be known to represent a full workflow. In these cases, the process could be represented in the form of a *work breakdown structure* (WBS) of tasks. If the dependency of one task on another is known, a default task sequence can be inferred (which in turn implies a default workflow). When more information is available, the process owner or the owner of a particular request can augment the information in the WBS to add conditional branches and determine specific task sequences.

Another form of expression for processes is that of an artifact-centric state model, rather than a flow model. A state model form of expression of processes is more intuitive in some cases because the sequence of tasks is predominantly determined by the states and transitions of the primary process artifact. For example, change management can be represented in the form of a state diagram for the primary process artifact, namely, the RFC.

The state model is an alternate representation where the focus is on the states, rather than on the sequence of actions that govern the transitions between the states (the activities and tasks). A state model representation can handle "loopbacks" more gracefully; for example, if an RFC is rejected, it could be sent back for reclassification, or it could be sent back for reassessment. These options can be represented as two different state transitions.

The PM designer has to determine the particular form of expression that is the most suitable for the domain. For example, a flow-based model may be most suitable for release management; whereas, incident management may be better handled with a combination of a flow-based model for the top level and an ad hoc model at the detailed level. Change management may be better handled with an artifact-centric state model, and asset management may be better handled with a WBS that represents the work involved in maintaining assets.

Generic and custom task types

To build process managers, the process manager designer has to determine the best approach for creating the user interface and business logic to help the user perform the task. A common approach to building process management applications in the industry is to build them as form-based applications. In such applications, the primary object is represented in the database, and the view associated with that object is created by use of a form metaphor. The users of the process application interact with the form by viewing, modifying, and filling in attributes. In all cases, they are limited to (or focused on) the particular set of attributes provided by the object and made visible in the form. This tends to work well when dealing with tasks that manage particular process artifacts, for example, collecting information, assessments, and approvals from different users for a particular change request. In such cases, the tasks in the process flow are essentially tasks that instruct the user to perform various actions with a form; for example, to approve a change request.

Using this approach, it is relatively easy to construct process flows with a set of generic task sequences, where the tasks have names corresponding to the steps in the process flow and the user interface for the task is a generic form user interface. We call such tasks *generic tasks*.

Generic work-management tasks themselves can have extended attributes for each task, enabling common variations of generic tasks to be addressed in this way. Beyond generic tasks, to support the more complex scenarios in PMs, it is also necessary to provide for custom task types that have their own task-specific GUIs and to display dynamic data based on tool interactions. For example, a user may need to interact with a tool as part of a process step. This kind of interaction is difficult to codify using forms technology. A more dynamic user interface is needed, one that can display the results of the interaction with the tool and allow the user to select or modify the information returned from the tool, thereby providing a more contextual, process-step-specific tool interaction. Such tasks are called *custom tasks*, because they have to be built with a specialized user interface and specialized business logic.

Process management tasks can thus support varying levels of sophistication and automation, especially when determining the appropriate manner of integration with systems management tools. This can range from fully manual tasks to fully automated tasks. Fully manual processes can be implemented simply by using a combination of generic tasks. The need for custom tasks increases for tasks that either have automation assistance or are completely automated.

The following are some of the ways that automation assistance for performing tasks can be provided:

- *Tool-linkage tasks*—manual tasks that carry a link (e.g., a URL [uniform resource locator]) to a tool that can be used to perform the task. Here the assistance to the user is the fact that the user does not have to search for the tool independently; instead, the specific instance of the tool and the home page of that tool are identified to the user.
- *Launch-in-context tasks*—assisted task implementations, in which context information from the current task is used to determine the target URL and the specific target location in a remote management tool. For example, when browsing a CI in a CMDB topology view, this task type would allow the user to launch to another tool (e.g., Tivoli Monitoring) to get the current monitoring details for the specific CI being worked on in the process management task.
- *Manual, tool-supported automation tasks*—a type of manual task performed by a person in a process flow, in which the person interacts with an external tool by means of the GUI provided by that process management task. One example is the IBM Tivoli Release Process Manager with a task-specific GUI to allow the user to browse the software depot in the Tivoli Provisioning Manager product to select the set of packages to import into the DSL. In this example, the process task implementation interfaces to the TPM tool through an integration module, makes queries of the TPM tool, and displays the results of the query in a GUI provided by the process task.
- *Fully automated tasks*—tasks performed in a completely automated manner as part of the process flow. For example, one of the tasks in a process might be to distribute software to a target machine. This step of the process flow could be done in a completely automated manner by a tool, based on data that was collected earlier in the process.

Thus, PM designers have to decide on the particular set of generic tasks and custom tasks that will have to be developed to support the particular end-to-end process automation scenarios that have been analyzed. PM implementations may also support multiple implementations for a single task (e.g., a manual implementation, an assisted implementation, or a fully automated implementation). In autonomic computing scenarios, the user is provided with mechanisms to delegate tasks to automation based on the confidence and comfort level of that particular user. This is analogous to the Windows Update feature, which can be configured to either automatically search for updates, download them and apply them automatically, simply search for updates and download them but not apply them, or notify the user of the updates without taking any other action. Similarly, PMs that support multiple task implementations may choose to provide the user the ability to choose the particular task implementation that is used for a particular step in the process flow.

To support any of the automated task types, tool integration modules have to be written. The integration modules map the generic task abstraction and data at the process level (e.g., distribute software CI to target CI) and translate it into the specific API (application programming interface) calls that are supported by a particular tool to implement that logical operation.

PM designers have to determine the level of automation in the custom task types supported and then create integration modules for each of the tools with which the process users need to interact.

OTHER CONSIDERATIONS

In this section, we discuss some additional issues related to process managers, namely, their integration with other process managers based on Web Services and their use in the real-time display of operational metrics and KPIs (key performance indicators).

External integration through Web Services

PMs need to interact with other PMs, as well as with external applications. To support such interactions, each PM must provide Web Services interfaces that allow external systems to invoke the functions of the PM. For example, the Change PM provides WSDL (Web Services Description Language) interfaces on the `ChangeManagement` port type and supports operations such as `CreateRFC`, `QueryRFCStatus`, `CancelRFC`, and `ModifyRFC`. The specific WSDL interfaces are unique to each PM and are governed by the semantics of the particular process domain.

Dashboards and reports

One of the primary reasons that organizations implement PMs is to improve organizational productivity. PMs must support this requirement by

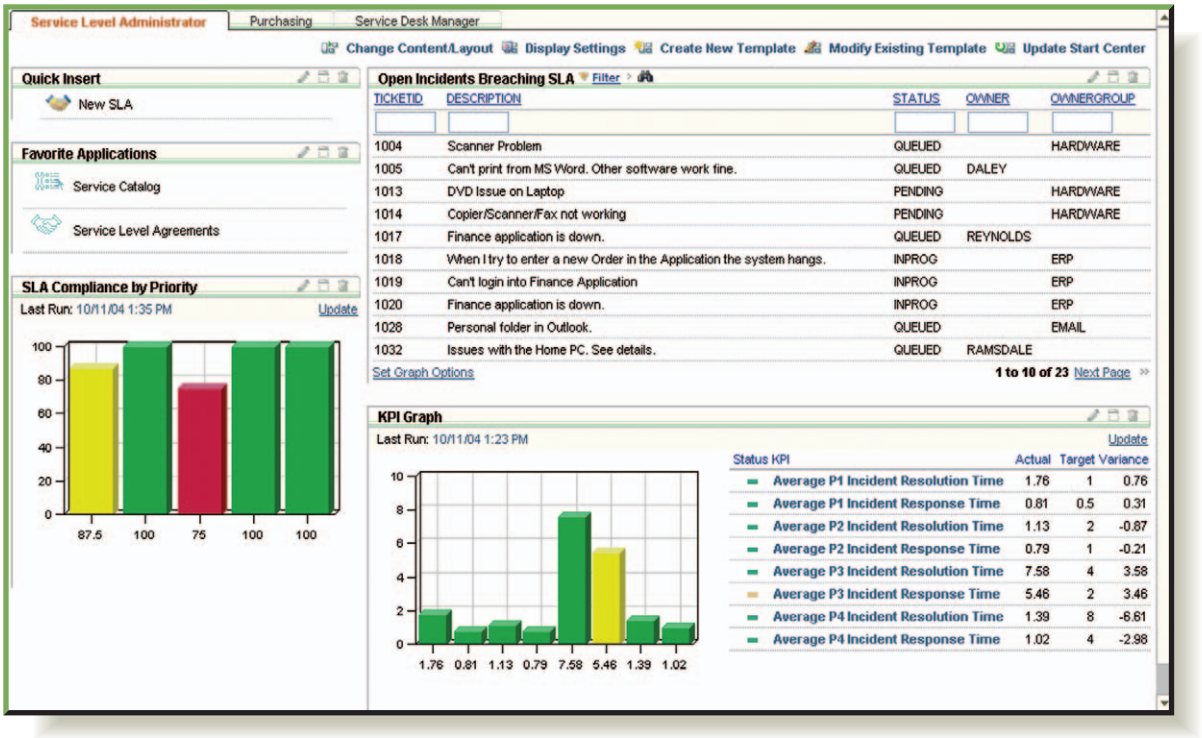


Figure 6
Example of a dashboard for a user role

providing their users with all the relevant information about operational metrics and KPIs. Operational metrics include information such as the number of RFCs in progress, the number of incidents that are open, the severity of the incidents, the end-to-end cycle time in handling incidents or RFCs, and the tasks that take the longest time in a process flow.

This information needs to be presented to users in real-time "dashboards" or in the form of reports. *Figure 6* shows an example of a dashboard view for a user logged into a PM. PMs need to enable users to create one or more dashboards based on the different roles that the user has. For example, in Figure 6, the user has the roles of service-level administrator, purchasing manager, and service-desk manager. Dashboards allow the user to perform view aggregation of the appropriate information based on the particular role that he or she is performing at a particular point in the process flow.

Although some of the information can be presented in real time by use of a dashboard view, a more detailed analysis is supported by the reporting component of the process manager.

Each PM collects many execution metrics. For example, every task is monitored during execution, and metrics like task duration, elapsed time, and number of pending tasks are maintained by the process management application (using information from the runtime platform on which the process manager is running). All of these metrics are available for eventual data mining and analysis of key performance indicators. Using these reports, organizations can understand problem areas and bottlenecks in their current processes and work on improving their processes for better organizational effectiveness and efficiency.

SUMMARY

The process of building a PM begins with understanding the best practices in a particular domain, performing scenario analysis to determine the appropriate business objects, request types, and process flows to build, and then going through the exercise of building the various components that make up the PM application. This paper covered many of the lessons we have learned in building PMs over the last few years; it also covered the architecture and design patterns about which PM

designers have to be knowledgeable when building new general-purpose process-management applications. Although the examples we have presented have been taken from the domain of IT management, the architectural patterns and design principles are applicable to other domains of process management as well.

ACKNOWLEDGMENTS

The author wishes to thank several members of the team who have provided inspiration and feedback over the years in defining and executing the strategy of building PMs. Special thanks to Vijay Aggarwal, Arnaud Mathieu, Dave Lindquist, Hari Madduri, Bala Rajaraman, Kyle Harding, Dave Calvert, Anthony Honaker, Praveen Hirsave, John Long, Mike Orr, Alan Lee, Ranjit Nayak, Adam Holey, Jeff Dean, Rob Goodling, and numerous others in the IBM Service Management organization. Several of the concepts discussed in this paper originated in an internal project called the ITIL Rapid Deployment Extensions (IRDE) project, for which the author worked jointly with Hari Madduri and Himanshu Desai.

CITED REFERENCES

1. H. Smith and P. Fingar, *Business Process Management: The Third Wave*, Meghan-Kiffer Press, Tampa, FL (2002).
2. F. R. Ricker and R. Kalakota, "Order Fulfillment: The Hidden Key to e-Commerce Success," *Supply Chain Management Review* **3**, No. 3, 60–70 (Fall 1999).
3. C. M. Hess and C. F. Kemerer, "Computerized Loan Origination Systems: An Industry Case Study of the Electronic Markets Hypothesis," *MIS Quarterly* **18**, No. 3, 251–275 (September 1994).
4. *Web Services Business Process Execution Language Version 2.0, Committee Draft*, OASIS (May 2006), http://www.oasis-open.org/committees/download.php/18714/wsbpel-specification-draft-May17.htm.
5. WebSphere Process Server, IBM Corporation, http://www.ibm.com/software/integration/wps.
6. F. Leymann and D. Roller, "Workflow-Based Applications," *IBM Systems Journal* **36**, No. 1, 102–123 (1997).
7. WebSphere Business Modeler Advanced, IBM Corporation, http://www.ibm.com/software/integration/wbimodeler/advanced.
8. WebSphere Business Monitor, IBM Corporation, http://www.ibm.com/software/integration/wbimonitor.
9. IT Infrastructure Library (ITIL), Office of Government Commerce, http://www.itil.co.uk.
10. itSMF International, http://www.itsmf.org.
11. COBIT, Information Systems Audit and Control Association (ISACA), http://www.isaca.org/cobit.htm.
12. Enhanced Telecom Operations (eTOM) Overview, TeleManagement Forum, http://www.tmforum.org/browse.aspx?catID=1648.
13. *The Sarbanes-Oxley Act of 2002 (House Resolution 3763)*, United States Congress (July 30, 2002), http://purl.access.gpo.gov/GPO/LPS22934.
14. HP Open View Service Center Overview and Features, Hewlett-Packard Development Company, http://www.openview.hp.com/products/ovsc/index.html.
15. BMC Software, Inc., http://www.bmc.com.
16. IBM Service Management, IBM Corporation, http://www-306.ibm.com/software/tivoli/governance/servicemanagement/gettingstarted.html.
17. D. Lindquist, H. Madduri, C. J. Paul, and B. Rajaraman, "IBM Service Management Architecture," *IBM Systems Journal* **46**, No. 3, 423–440 (2007, this issue).
18. IBM Tivoli Release Process Manager, IBM Corporation, http://www.ibm.com/software/tivoli/products/release-process-mgr/.
19. IBM Tivoli Storage Process Manager, IBM Corporation, http://www.ibm.com/software/tivoli/products/storage-process-mgr/.
20. SAP United States, Enterprise Applications/Business Software Systems, http://www.sap.com/usa/solutions/index.epx.
21. Siebel Customer Relationship Management Applications, http://www.oracle.com/applications/crm/siebel/index.html.
22. List of BPEL Engines, Wikipedia, http://en.wikipedia.org/wiki/List_of_BPEL_engines.
23. Rational Method Composer, IBM Corporation, http://www.ibm.com/software/awdtools/rmc/.
24. *Information Technology—Service Management—Part 1: Specification*, International Organization for Standardization (ISO) (2005), http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=41332&ICS1=35&ICS2=20&ICS3=.
25. CMMI, Carnegie Mellon University, Software Engineering Institute (2007), http://www.sei.cmu.edu/cmmi/.
26. IBM Tivoli Unified Process, IBM Corporation, http://www-306.ibm.com/software/tivoli/governance/servicemanagement/itup/tool.html.
27. WebSphere Integration Developer, IBM Corporation, http://www.ibm.com/software/integration/wid/.
28. A. Ganek and K. Kloeckner, "An Overview of IBM Service Management," *IBM Systems Journal* **46**, No. 3, 375–385 (this issue, 2007).

Accepted for publication March 26, 2007.
Published online July 3, 2007.

Chakalamattam Jos (C. J.) Paul
IBM Software Group, Tivoli, 11501 Burnet Road, IBM Tivoli Software, Austin, TX 78758 (cjpaul@us.ibm.com). Dr. Paul is the lead architect for Process Managers in IBM Tivoli and guides the architecture and design of the family of Process

Managers that are 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. ■