

Realtime
publishers

"Leading the Conversation"

The Definitive Guide[™] To

Service-Oriented Systems Management

Dan Sullivan

Chapter 5: Implementing System Management Services, Part 1: Deploying Service Support	92
Elements of Service Support.....	93
Interdependent Service Support Processes	93
Automated Configuration Management	94
Data Collection Procedures.....	94
Centralized Data Repository	95
Process Flow Support	95
Information Retrieval.....	96
Incident Management.....	98
Characteristics of Incidents.....	98
Severity	98
Assets	98
Personnel.....	99
Resolution Method.....	99
Incident Types.....	99
Resolving Incidents.....	101
Problem Management	102
Trend Analysis	103
Configuration Management	103
Planning	103
Identification	104
Control	105
Status Accounting	105
Verification and Audit	105
Change Management	106
Ripple Effects of Change	106
Change Controls.....	107
Requests for Change	107
Change Advisory Board.....	108
Release Management	108
Planning Releases	109
Testing and Verifying Releases	109
Software Testing	110

Data Migration Testing110

Integration Testing110

Software Distributions111

Communications and Training.....112

Summary112

Copyright Statement

© 2006 Realtimepublishers.com, Inc. All rights reserved. This site contains materials that have been created, developed, or commissioned by, and published with the permission of, Realtimepublishers.com, Inc. (the “Materials”) and this site and any such Materials are protected by international copyright and trademark laws.

THE MATERIALS ARE PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. The Materials are subject to change without notice and do not represent a commitment on the part of Realtimepublishers.com, Inc or its web site sponsors. In no event shall Realtimepublishers.com, Inc. or its web site sponsors be held liable for technical or editorial errors or omissions contained in the Materials, including without limitation, for any direct, indirect, incidental, special, exemplary or consequential damages whatsoever resulting from the use of any information contained in the Materials.

The Materials (including but not limited to the text, images, audio, and/or video) may not be copied, reproduced, republished, uploaded, posted, transmitted, or distributed in any way, in whole or in part, except that one copy may be downloaded for your personal, non-commercial use on a single computer. In connection with such use, you may not modify or obscure any copyright or other proprietary notice.

The Materials may contain trademarks, services marks and logos that are the property of third parties. You are not permitted to use these trademarks, services marks or logos without prior written consent of such third parties.

Realtimepublishers.com and the Realtimepublishers logo are registered in the US Patent & Trademark Office. All other product or service names are the property of their respective owners.

If you have any questions about these terms, or if you would like information about licensing materials from Realtimepublishers.com, please contact us via e-mail at info@realtimepublishers.com.

[**Editor's Note:** This eBook was downloaded from Realtime Nexus—The Digital Library. All leading technology guides from Realtimepublishers can be found at <http://nexus.realtimepublishers.com>.]

Chapter 5: Implementing System Management Services, Part 1: Deploying Service Support

Much of the work in systems management is service support—keeping devices and applications functioning and ensuring that they continue to meet the changing needs of the organization. This task entails managing changes as new assets are added and others are retired; reconfiguring systems in response to changes in the infrastructure, such as growing demands for network bandwidth; and releasing new versions of applications to geographically distributed users. Service support is especially challenging because of the breadth of services that are typically supported by IT operations and the depth of detailed information required for service support.

The breadth of operations, from upgrading operating systems (OSs) and reconfiguring routers to planning software releases and responding to security incidents, can be labor intensive. For example, upgrading the OS on one desktop computer might take one hour in a simple case. Coordinating times to install the upgrade with users and dealing with unexpected consequences of the change add to that time.

Ensuring the Quality of Service (QoS) delivery depends upon detailed information about the state of devices and processes running on those devices. A systems manager cannot simply install a new application or upgrade an existing application without understanding how the system is currently used. For example, a Java application server may depend upon one version of the Java runtime environment (JRE), but another application, about to be installed on the same server, requires a different version of the same runtime environment. The systems manager cannot uninstall one version of the runtime environment and replace it with another without disrupting the application server operations.

Clearly, to be effective and efficient, service delivery operations must be built on a foundation of well-defined processes and, ideally, automated operations. The previous chapter introduced the configuration management database (CMDB) as a central component of service-oriented management. This chapter builds on that with a discussion of processes that leverage the CMDB. In particular, this chapter will cover the common characteristics of the multiple elements of service support, as well as details about:

- Incident management
- Problem management
- Configuration management
- Change management
- Release management

The chapter concludes with a discussion of the unifying elements of service-oriented management with respect to service delivery.

Elements of Service Support

Service support is about responding to change. The needs of users change. Configurations change. Unexpected events occur. The specific details of these changes will vary, but how systems managers respond should not. A set of well-defined processes are at the core of service-oriented systems management. Those processes—incident management, problem management, configuration management, change management, and release management—are discussed in detail later in this chapter. In this section, the focus is on the shared attributes and interdependencies of these processes.

Interdependent Service Support Processes

Service support processes have different specific goals depending on the area of service delivery they address (see Figure 5.1). For example, in incident management, something has disrupted the normal flow of operations, and the goal is to restore normal services as soon as possible. Problem management, in contrast, takes a more holistic approach and attempts to prevent incidents by detecting patterns in incidents and identifying root causes. Even with their different goals, service support processes are highly interdependent.

In some cases, those root causes could be the result of improperly configured devices, in which case, configuration management processes must be examined. Were they applied properly? Is there a deficiency in the process that allowed a flawed configuration to enter production service? Perhaps the correct configuration specifications had been defined but they were not properly installed; in that case, the release management procedures require review.

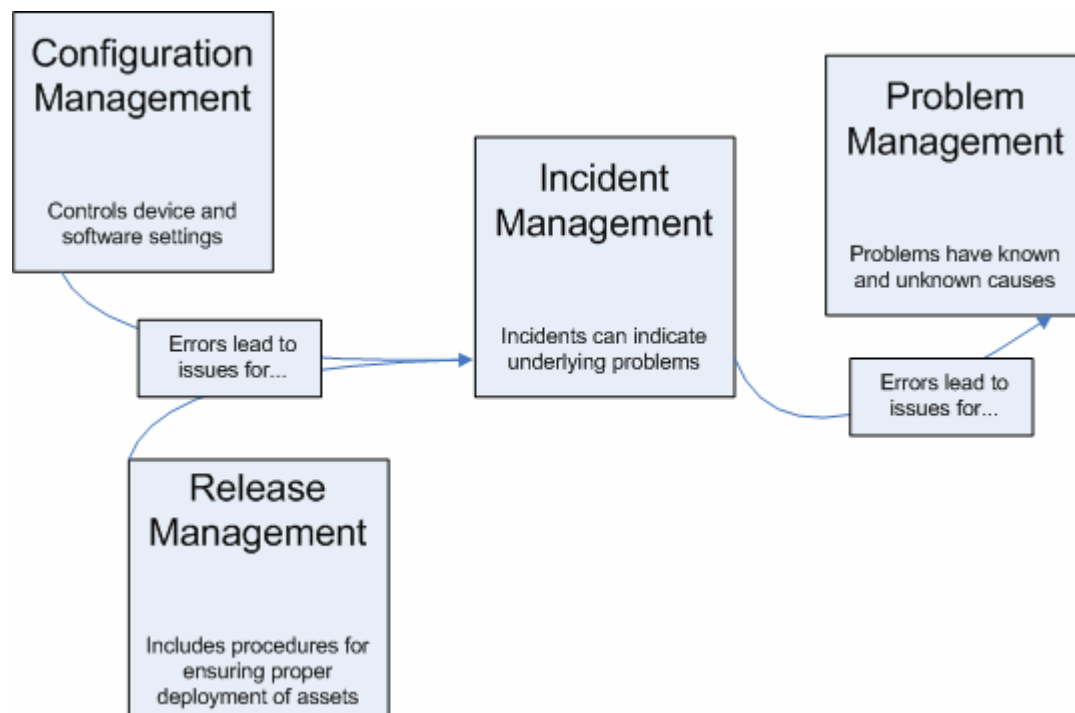



Figure 5.1: Service support processes are interdependent and can support or trigger other each other.

It is clear from these simple examples that information relevant to one process can be vital to the proper implementation of other processes. It is also clear that errors in one set of procedures can have ripple effects that cause other processes to be activated. For both of these reasons, automated configuration management services can improve service support.

Automated Configuration Management

The objective of automated configuration management is twofold. First, it is to obtain and maintain information about the state of devices and applications deployed throughout the IT infrastructure. The second objective is to support multiple IT operations, especially service support.

 Automated configuration management is a mechanism that supports all parts of service support, not just traditional configuration management. This mechanism should not be confused with early configuration management tools that provided limited information and provide little support for related service support operations.

To meet these objectives, automated configuration management applications use several modules, including:

- Agents for collecting configuration and other status information
- Centralized data repository
- Process flow support
- Information retrieval

Together, these modules provide the core services of automated configuration management.

Data Collection Procedures

Data is gathered from devices using agents, or applications that collect information locally and transmit it to a central repository. These agents should be relatively lightweight and autonomous; once installed and configured, they should require little systems manager intervention.

The configuration processes entail setting a number of characteristics, including:

- Data collection policies
- Frequency of data collection
- Data transmission information
- Authentication mechanism

The data collection policies define what information is gathered and how frequently it is transmitted. The information gathered can include local security policy settings, storage utilization, significant system events, and other audit-related details.

The frequency of data collection will determine how often the agent sends information to the central repository. There is a tradeoff with this setting. Devices that frequently update the central repository are less likely to have outdated data, but the data collection process places additional demands that can adversely impact the performance of other applications.

When depending on agents, it is important for the central repository to accept data only from authenticated agents. Distributed applications such as these are vulnerable to spoofing—that is, an attacker or an attacker’s program pretending to be the real agent. An attacker, for example, might want to cover his or her tracks by sending false information about failed login attempts or the amount of disk space in use. By using cryptographic techniques, such as digitally signing all transmissions, the repository can significantly reduce the chance of attacks. (See the sidebar, Digital Signatures, for details about this security measure).

Centralized Data Repository

A centralized data repository for configuration management is one that supports multiple functions related to service delivery, including managing configurations, which, in turn, support both service delivery and security enforcement. Although the basic role of the repository is to answer queries about the state of devices, it must be designed to support queries from multiple domains. For example, from an incident management perspective, the database might be queried about the software installed on a particular device and the dependencies between those applications. This is useful in cases in which a newly installed application is not working properly but works correctly on other similarly configured clients. In such a case, one of the first questions to answer is: What are the differences between clients with a working installation of the application and clients without?

In the case of problem management, support personnel may discover that a particular version of a browser add-in causes parts of an application interface to fail. They may also find that rolling back to an earlier version of the add-in resolves the problem. In this case, the configuration database could be used to determine all devices that have both the problematic plug and need to run the thin-client application. After the correct plug-in has been deployed, the database can be queried to verify installation (assuming agents have updated the repository). These are relatively simple examples; other more complex issues may require multi-step procedures.

Process Flow Support

Configuring IT devices often entails dependencies between components. Mechanisms for supporting process flow can help control procedures that must be aware of those dependencies.

Consider requirements for rolling out a Web-based application that uses Java-based technologies in clients’ browsers. In addition to updating client browsers with the latest security patches, the release requires that the JRE is installed to a particular revision level. Once the browser is patched and the JRE installed, a plug-in must be installed within the browser as well. Each of these steps must be done in sequence and if one step fails, the succeeding steps should not occur. The results of the installation must be verifiable.

A process flow engine within a configuration management system could meet these requirements if it supports:

- Ordered deployment of modules
- Tests for success of each step
- Conditional processing—for example, if the browser does not contain a particular patch, it is installed; otherwise, it is not
- Detail logs of each step

Logged information about the deployment process should be available by querying the CMDB.

Information Retrieval

Information retrieval sounds trivial—you simply want to display data that is stored in a database. What is not trivial is precisely specifying what data it is that you want displayed. At one end of the information retrieval spectrum, there are query languages used by database developers and the occasional power user. Even for relatively simple queries, this is not a reasonable tool for most users. Consider the following query: a systems manager wants to list all resource associations, the associated resource type, the name of the resource, and a brief description, sorted by resource type. The corresponding database query would look something like (the details depend on the database structure, but the example holds for a typical normalized relational database):

```
SELECT
    ra.resource_assoc_name,
    rt.resource_assoc_type_name,
    rt.resource_type_name,
    r.resource_name
    r.resource_descr
FROM
    resources r,
    resource_type rt,
    resource_associations ra
WHERE
    r.resource_id = ra.resource_id
AND
    r.resource_type_id = rt.resource_type_id
ORDER BY
    rt.resource_type_name
```

Query languages are not practical tools for working with CMDBs—they require an understanding of the underlying data model and knowledge of the database query language, typically a variation on ANSI standard SQL. However, query languages are quite flexible and with the right query, one can find anything that is in the database.

Static reports lie at the other end of the information retrieval spectrum. They require no knowledge of the implementation details of the database, but they are limited in their usefulness. Static reports provide information about a limited amount of data and typically represent designers and developers' best guess at what information a systems manager will need.

Between the two extremes lies parameterized reports. They provide some of the flexibility of query languages along with some of the ease of use of static reports. Properly configured, these reports can help guide users to the information they need (see Figure 5.2 for an example).

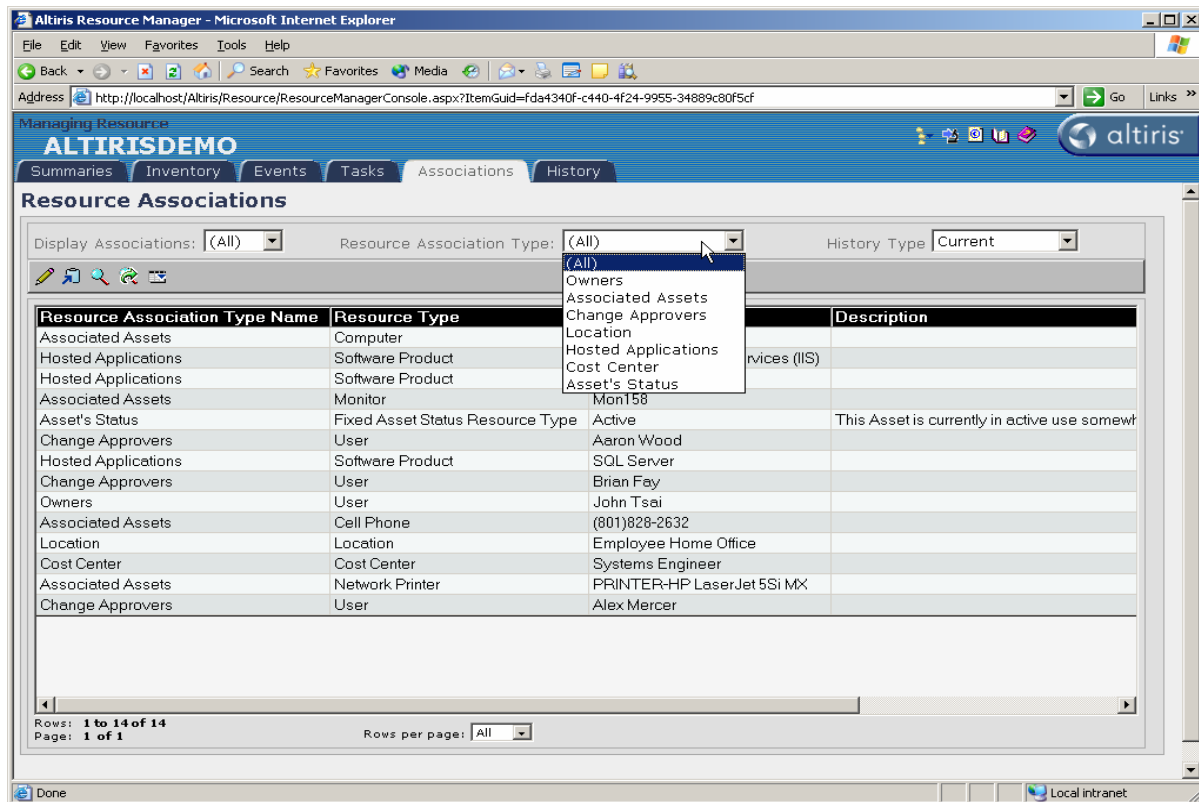


Figure 5.2: Information retrieval from complex data structures should use a combination of search and guided querying.

Automated configuration management tools provide several mechanisms important for efficient service support, including a centralized data repository, automated data collection, support for process flow, and flexible reporting. The following sections describe how automated configuration management can support the particular requirements of several service support areas.

Incident Management

Incidents are events outside of the normal operations that disrupt those operational processes. An incident can be a relatively minor event, such as running out of disk space on a desktop machine, or a major disruption, such as a breach of database security and the loss of private and confidential customer information. Incident management is a set policies and processes for responding to incidents, the goals of which are to:

- Restore normal operations as quickly as possible
- Track information about incidents for further analysis
- Support problem management by analyzing patterns of incidents

Incident management begins with defining what constitutes an incident, categorizing those incidents, and measuring their occurrences.

Characteristics of Incidents

Something as generalized as “any event outside of normal operations” covers quite a large space of possible events. By focusing on just those that are so disruptive that they cause a call to the Help desk or other IT support services, you can limit the discussion to a manageable domain. Within this domain of incidents, you can categorize incidents by several characteristics:

- Cause of problem
- Severity
- Asset or assets causing the incident
- Role of personnel experiencing disruption
- Resolution method

The cause of problems covers a wide range of topics.

Severity

Incidents should be categorized by severity; at the very least a three-point scale of minor, moderately severe, and severe should be used. For each level of severity, IT organizations should define acceptable resolution times, escalation procedures, and reporting procedures. For example, minor incidents, such as password resets, should not consume too much time or resources from the Help desk. A security breach, however, should immediately escalate, trigger reporting to management and executives, and require rapid resolution.

Assets

The asset or assets causing an incident are important dimensions for tracking incident trends. If a particular version of desktop application is causing an inordinate number of support calls, IT managers should be able to detect this during problem management procedures. (There is more information about problem management later in this chapter.)

Personnel

Just as assets involved in incidents should be tracked, so should the users encountering the disruptions. If a large number of personnel from a single department are generating a large number of Help desk calls, there might be a problem with training or an application specific to that department.

Resolution Method

The method for resolving an incident should also be tracked. This data can help determine guidelines for selecting the appropriate response to an incident. For example, data about resolution methods reveal that most OS problems that require more than 2 hours to solve eventually require reinstallation. Given that, a support desk policy is instituted requiring that OS errors that cannot be resolved within 2 hours will be addressed by formatting the OS drive and restoring it from an image backup. These characteristics are especially useful when measuring incident rates and analyzing trends by these characteristics.

Incident Types


Defining the cause of a problem can be more difficult than it seems at first because there are sometimes multiple pre-conditions that must be in place for an incident to occur. Consider a few examples. Password resets are one of the most common incidents reported to Help desks. The causes of this type of incident include users allowing passwords to expire and forgetting passwords—especially when users are expected to remember passwords to multiple systems while not re-using passwords. All of these causes can factor in a single password reset incident.

In another example, an employee is saving a document to a network drive when the save operation fails. An error message is displayed stating the network drive cannot be found. Because the employee had been saving the document regularly, something must have occurred since the last save operation. After the user has contacted the service desk, the service desk technician tests several possible causes and determines that the problem is a failed network interface device. In this case, determining the exact cause of the failure is not relevant unless the problem occurs repeatedly; hardware has well-known mean times between failures (MTBFs) and further root cause analysis is not likely to help reduce these types of incidents.

The final example is more complex. A security breach results in a large number of customer account and credit card numbers being exposed to attackers. The causes could include:

- Improperly configured firewalls that allow traffic on a port that should have been closed
- An un-patched database listener (a program that accepts requests to connect to the database) that is vulnerable to known attacks
- Access controls within the database that do not adequately limit read access to sensitive data
- Vulnerability in a database management system that allows for escalation of privileges
- Lax OS privileges that allow execute privileges on database administration tools
- Poorly designed applications that use over privileged database accounts

A database breach is a case in which a series of vulnerabilities must be in place for a successful attack to occur. Had one of the vulnerabilities been compensated for with adequate countermeasures, the attack would not have occurred as it did. For example, had the access controls on database tables and views been sufficiently restrictive, the attacker could not query the sensitive data even though he or she had made it through network, OS, and database authentication security measures.

 Information security is often compromised by a weak link; however, one effective countermeasure can stop an attack that has exploited a number of other weaknesses. A best practice in security, known as *defense in depth*, deploys multiple countermeasures to protect assets even if, in theory, only one is needed. Security practitioners know that no single measure is perfect and multiple countermeasures are needed to reduce the risk of information theft and other threats.

The general categories of incident causes that cut across these examples include:

- Improper documentation
- Insufficient user training
- Configuration errors
- Previously unknown bug
- Known but un-patched vulnerabilities
- Unexpected changes in operating loads
- User error

Determining the cause of incidents is essential to understand both how to resolve the problems and how many resources to commit to reduce the likelihood of those problems in the future.

Resolving Incidents


Of all the topics in service support, the most time could be spent on resolving incidents; in fact, it could be the topic of a very long book. The problem with resolving incidents is that there are so many types and each can require a customized response. In some ways, resolving incidents is like cooking—there is a different recipe for every dish, and there is a different response to every incident. At the same time, general principals can be found that apply to a broad range of challenges, whether culinary or technical.

The general principals for resolving incidents include:

- The time, effort, and resources committed to incident resolution must be commensurate with the impact of the incident.
- Responses should be formalized with well-defined procedures that are more frameworks than strict, precise sets of steps. Formulating such procedures would be too time-consuming to be practical.
- All incidents and the response should be documented. In some cases, this can be as trivial as incrementing a count of simple incidents, such as password resets, or as complex as a detailed report describing a security breach.
- As with other service support operations, coordinate incident resolution information with other asset information.

Consider examples from the extremes of resolving incidents: Password resets are one of the simplest types of incidents to resolve. Many organizations now use self-service methods to address them. One could attempt to drive down the number of password resets, but after a certain point, the economics do not justify the effort to do so because the marginal cost of resetting a password with a self-service system is small. As the next section on trend analysis will show, password vulnerabilities could become a factor in broader security management issues in which the costs of poor password management grow much higher.

Security incidents are some of the most costly. According to the FBI/Computer Security Institute (CSI) Computer Crime and Security Survey, 639 respondents reported a total loss of almost \$43 million due to virus attacks and more than \$31 million due to unauthorized access. Individual incidents can be extremely costly. For example, 40 million credit card accounts were compromised at CardSystems Solutions, a credit card processor, causing it to lose major credit card customers.

 For more information about the CardSystems Solutions breach, see Clint Boulton's "MasterCard: 40M Credit Card Accounts Exposed" at <http://www.crime-research.org/news/28.06.2005/1321/>. The FBI/Computer Security Institute 2005 Computer Crime and Security Survey is available at http://i.cmpnet.com/gocsi/db_area/pdfs/fbi/FBI2005.pdf.

Resolving incidents requires detailed information, whether one is dealing with password resets or security breaches. A centralized repository of configuration information is especially helpful when the incident is caused, in part, to hardware, software, or system configurations.

Problem Management

Problem management is focused on reducing incidents and their impact on an organization's operations. Problem management and incident management, although tightly coupled, differ in several ways:

- A problem is the underlying cause for multiple disruptions; an incident is one of those disruptions.
- Problem management addresses the underlying cause of multiple incidents; incident management entails responding to an instance of disrupted operations caused by a problem.
- Problem management attempts to detect and address root causes of problems; incident management attempts to restore normal operating functions, possibly without fully correcting the underlying cause.

Problem management depends on data from multiple incidents, so a CMDB and incident repository can support the investigation and analysis of root causes. For example, if an end user application repeatedly crashes on some but not all client devices, the CMDB can be used to determine what the affected systems have in common that are not found in the unaffected devices.

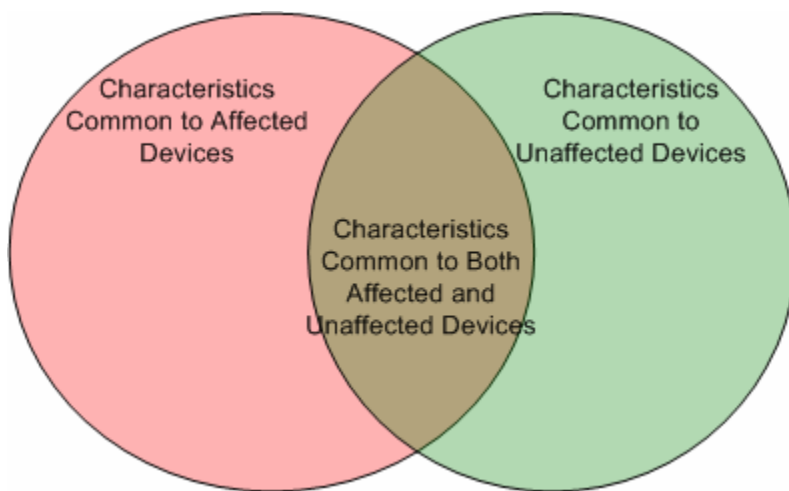


Figure 5.3: CMDBs can help to rapidly identify common characteristics of devices affected by an incident, thus supporting root cause analysis and problem management.

Once the cause of a problem is identified and a solution developed, the problem and solution should be documented for future reference. Even if the identical problem is not likely to occur again—for example, all servers are patched for a known vulnerability—the solution description may help to solve other somewhat similar problems.

Trend Analysis

Another part of problem management, and closely related to incident management, is trend analysis. The function of trend analysis is to determine the frequency of particular types of problems and determine which, if any, incident types are increasing. Trend analysis can lead to introducing new methods or devices. For example:

- The increasing number of password resets, coupled with the cost of staffing Help desks, can create a cost justification for a self-service password reset.
- Rapid growth in email storage requirements may justify the use of a network appliance to filter spam.
- Discovery of an increasing number of conflicts between newly deployed applications and legacy applications can lead to changes in software testing methodology.

Trend analysis in itself does not solve problems but identifies categories of problems that are growing in severity or frequency. A general problem that can have ripple effects throughout an IT infrastructure is errors in configuration management.

Configuration Management


Configuration management is the process of controlling changes to device configurations in an IT environment. There are five basic operations in configuration management:

- Planning
- Identification
- Control
- Status accounting
- Verification and audit


Together, these operations provide the means to control the establishment and maintenance of device configurations.

Planning

Planning within configuration management is similar to other IT operations; that is, the focus is on setting an overall strategy, defining the policies and procedures necessary to implement that strategy, and identifying configuration items that should be tracked within the CMDB. The configuration management strategy defines the scope and objectives of the configuration management process. For example, the scope of a typical plan includes all managed devices within an organization; the objectives include maintaining the availability and integrity of devices, ensuring efficient use of resources, and minimizing maintenance and training costs.

 Managed devices are those that are under the control of an organization and function within the IT infrastructure; unmanaged devices function within the IT infrastructure but are uncontrolled by the IT department. Examples of unmanaged devices include servers used by business partners and desktops used by customers to access online services.

The planning process also defines roles and responsibilities. A single device may be maintained by several roles. A server, for example, may be the responsibility of a systems manager who is responsible for the OS and access controls, a network administrator who is responsible for configuring network hardware and protocols, and an application administrator who maintains services provided by the server. The CMDB is used across service support operations, but its function and maintenance fall under the scope of configuration management planning.

 It should be noted that configuration management planning is not a one-time event. These plans are typically subject to change as business and organizational requirements change. A comprehensive review of configuration management plans is recommended twice year.

Although the planning process focuses on the overall configuration management process, the identification process addresses the details of the operation.

Identification

Any entity tracked by configuration management is known as a configuration item (CI). Several characteristics of configuration items are recorded:

- Name and description
- Owner of item
- Relationships to other items
- Versions
- Unique identifiers

It is important to identify CIs to the level of independent change. For example, if laptops are treated as a single unit and hard drives are not moved among laptops, there is no need to track the hard drives independently of the laptop. However, optical drives used for backups and moved among servers should be managed as distinct devices.

Control

The control process ensures that all configuration items are properly identified, their information is recorded in the CMDB, and any changes are done in accordance with change management procedures. (Change control is discussed in detail later.)

Status Accounting

Status accounting is the process of recording state changes to a configuration item. The most common states are:


- On order
- Received, pending testing
- Under test
- Installed to production
- Under repair
- Disposed

All state changes should be recorded so that the CMDB always has an accurate representation of the IT infrastructure. This information is also useful for problem management, especially for detecting devices with high incidents of repair or long repair periods.

Verification and Audit

During verification and audit, the contents of the CMDB are compared with the physical configuration items to ensure that information about them is correctly recorded. Documentation about changes to configuration items should also be verified during audits.

Configuration management is an ongoing operation. Some processes, such as planning and verification, occur at regular intervals. The other processes are continuous.

 The Configuration Management II Community at <http://www.cmcommunity.com/> provides resources, forums, and other material related to configuration management.

Change Management

Change management is the process of controlling modifications to configuration items so as to minimize incidents that disrupt normal operations. The reason change management is so important is that one change can have ripple effects through multiple other assets (see Figure 5.4).

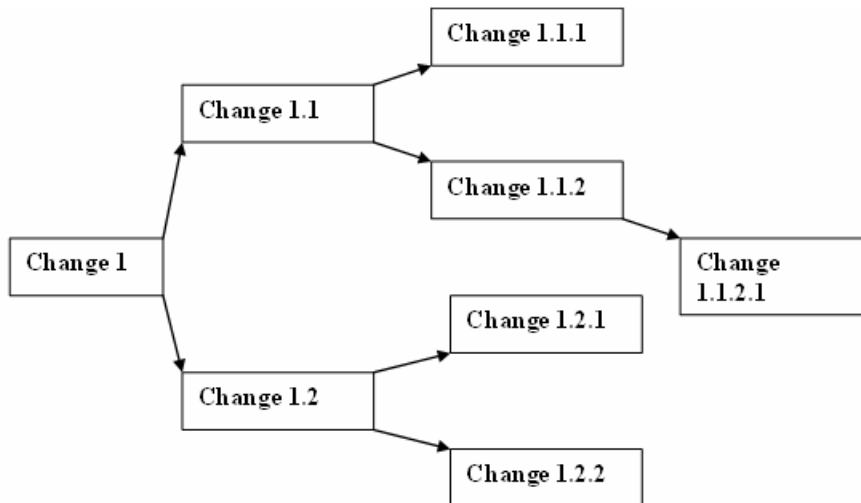



Figure 5.4: Changes in one configuration item can have ripple effects through other items.

Ripple Effects of Change

Consider a change to a hypothetical management reporting system. Until now, managers have received static reports in Adobe PDF format once a week summarizing the business activity of their units. The IT department is deploying an interactive reporting system. Here are some of the changes related to the new system:

- Client software must be installed on power users' desktops and laptops; basic users will use a Web-based system to retrieve reports.
- The new client software for power users will have to be added to the patch management process to ensure clients receive all relevant security updates.
- Middleware software, including a downloadable applet, will have to be installed on Web servers.
- Firewall rules will be modified to allow traffic on a newly opened port to the Web component of the reporting system.
- New groups and roles will be added to the database access control system to allow managers to retrieve information using their network login ID.
- Additional database servers may be required to support the additional load of ad hoc querying.
- Failover servers may be added to ensure managers access to the database if a Web server should fail.

Clearly, what appears at first to be a software change can quickly propagate ripple effects to other software components, hardware devices, and network settings.

 Large numbers of emergency change requests is an indication of failures in other processes, such as planning, testing, patch management, and security management.

Change Controls

Formal change control procedures are one way to ensure that the effects of a change are understood before the change is implemented. Formal methods are often based on a standardized change request mechanism and a change review board.

Requests for Change

A request for change is a documented description of a change to the IT infrastructure. A change can be as simple as installing a new version of a word processor on a user's laptop to deploying a new application for hundreds of users. Regardless of the complexity, several characteristics of changes should be documented:

- Originator of request for change
- Configuration item to change
- Implementation plans
- Back out plans
- Type of change, such as a change in requirement, bug workaround, additional hardware for increased demand, and so on
- Reason for the change, such as compliance, policy change, defect, new business requirement
- Priority, which can include emergency, urgent, or routine
- Detailed description of change
- Estimated time and resources required to implement
- Impact analysis for complex changes
- Approvals

Some of these items, such as the change description and the reason for change, are defined by the user or department making the request; others, such as the time and resource estimate and impact analysis, are determined by technical staff. The approvals required will depend on the priority and the complexity of the change. A relatively simple change, such as a desktop application installation, may require line manager approval, but a major software release will require approval from several managers, both on the business and technical sides of the organization.

Change Advisory Board

A change advisory board (CAB), sometimes called a change control board, is responsible for reviewing complex requests for changes. A CAB should include managers responsible for business operations as well as technical staff familiar with systems administration, network operations, security, and software development and management. The purpose of the CAB is to provide visibility to changes before they are made—not to slow the change process.

CABs are sometimes seen as bureaucratic road blocks to getting work done—they should not be. Their role is to provide a checkpoint to ensure that the implications of changes are thought through as completely as possible. Correcting a problem with a proposed change is much easier and less expensive when done at the planning stages than after the modification has been made. In organizations with mature, functional IT operations, change is managed as a formal process. There is no avoiding change, but you can determine how we deal with it.

After changes have been reviewed, revised and approved, the next step is implementation. This domain is addressed by release management practices.

Release Management

Release management is a demanding operation. The goal of release management is to preserve the integrity and availability of production systems while deploying new software and hardware. Several processes are included under the umbrella of release management:

- Planning software and hardware releases
- Testing releases
- Developing software distribution procedures
- Coordinating communications and training about releases

Release management is the bridge that moves assets from development into production.

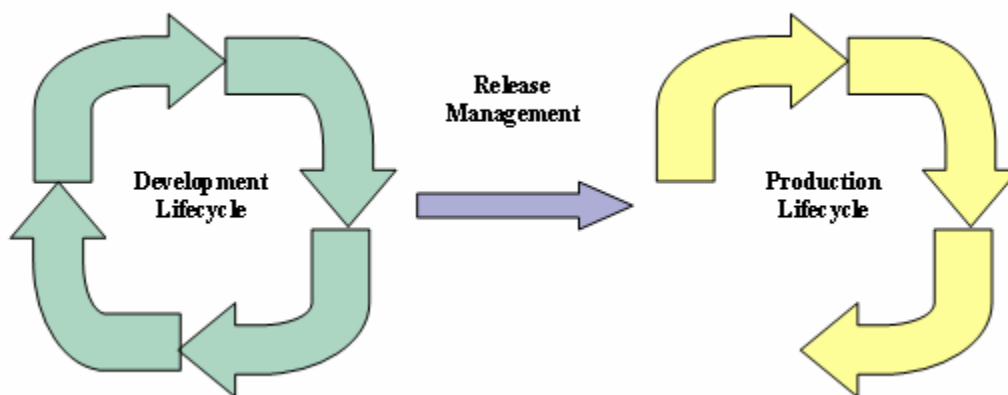



Figure 5.5: Release management is the bridge between two high-level IT life cycles: development and production.

Planning Releases

Planning releases is often the most time-consuming area of release management because there are so many factors that must be taken into consideration. For example, when deploying a new sales support system, the release managers must address:

- How to distribute client software to all current users
- How to migrate data from the current applications database to the new database with minimal disruption to database access
- How to verify the correct migration of data
- How to uninstall and decommission the applications replaced by the new system
- Verifying all change control approval are secured

Each of these issues breaks down into a series of more granular tasks. Consider distributing client software. Release managers must account for variations in OSs and patch levels of client devices, the need for administrative rights to update the registry if software is installed, and the possibility of conflicts or missing software on clients.

 One of the often-discussed advantages of Web-based applications is that client software does not need to be permanently installed. This is true for the most part, but some software is still required to support Web applications, including browsers, browser helper objects (BHOs), plug-ins, and, in some cases, a JRE. The supporting software is subject to some of the same constraints and limitations as client/server software—they sometimes require administrative privileges to install, they must be patched as needed to maintain security, and they are subject to their own upgrade life cycles. Web-based applications may ease some of the burdens associated with release management, but they do not eliminate them.

Testing and Verifying Releases

Release managers can play an important part in the testing phase of the software development life cycle; the key areas for testing and verification are:

- Software testing
- Data migration testing
- Integration testing

In each case, the testing process should constitute the final test and primary verification that the newly deployed applications operate as expected in the production environment.

Software Testing

It goes without saying that software should be thoroughly tested before it is released. In the ideal world, software developers work in the development environment and deploy their code to a testing and quality assurance environment that is identical to the production environment. It is in the test environment that integrated module testing and client acceptance testing is performed. This is not always possible. Large production environments may not be duplicated in test environments because of cost and other practical limitations. It is especially important in these cases that release managers work closely with software developers.

With responsibility for deploying software, release managers can provide valuable implementation details about the production environment that developers should test. For example, release managers will have information about the types of client devices and the types of network connectivity supported as well as other applications that may coexist with the system under development. Release managers may need to address data migration issues as well.

Data Migration Testing

In addition to supporting software developers on application testing and quality assurance processes, release managers may also have to support database administrators who are responsible for migrating data between applications. When the release of a new application entails shutting down one system, exporting data, transforming it to a new data model, and importing it into the new system, release managers will share responsibility for ensuring the data is extracted, transformed, and loaded correctly. Again, this process should be thoroughly tested prior to release, but realistic data loads are not always possible in test environments.

Integration Testing

Integration testing is the process of testing the flow of processing across different applications that support an operation. For example, an order processing system may send data to business partners' order fulfillment system, which then sends data to a billing system and an inventory management system. Certainly these would have been tested prior to deployment, but real-world conditions can vary and uncommon events can cause problems. For example, spikes in network traffic can increase the number of server response timeouts forcing an unacceptable number of transactions to rollback. In this case, it is not that the systems have a bug that is disrupting operations, but that the expected QoS levels are not maintained. Testing and verifying software functions, data migration, and integrated services can be easily overlooked as "someone else's job," but release managers have to share some of this responsibility.

Software Distributions

Software distribution entails several planning steps. At the aggregate level, release managers must determine whether a phased release is warranted, and if so, which users will be included in each phase. Phases can be based on several characteristics, including:

- Organizational unit
- Geographic distribution
- Role within the organization
- Target device

When deploying new software or major upgrades, a pilot group often receives the software first. This deployment method limits the risks associated with the release. (Even with extensive testing and evaluation, unexpected complications can occur—especially with end users' response to a new application).

When distributing software, several factors must be considered:

- Will all clients receive the same version of the application? Slightly different versions may be required for Windows 2000 (Win2K) clients and Windows XP clients.
- Will all clients receive the same set of modules? If a new business intelligence application is to be deployed, power users may need full functionality of an ad hoc query tool and analysis application, while managers and executives may require only summary reports and basic drill-down capability.
- How will installation recover from errors or failure? Downloads can fail and need to be restarted. There may be a power failure during the installation process. Disk drives can run out of space. In some cases, the process can restart without administrator intervention (for example, when the power is restored) but not in other cases (such as when disk space must be freed).
- How will the installation be verified? Depending on the regulations and policies governing IT operations, differing levels of verification may be required. At the very least, the CMDB must be updated with basic information about the changes.

Software distribution is the heart of release management, but the ancillary process of communication and training are also important.

Communications and Training

The goal of communication in release management is to make clear to all stakeholders the schedule and impact of releases. This is the responsibility of release managers.

Training users and support personnel on the released system is not the responsibility of release managers, but both training and release managers should coordinate their activities. When training occurs too far in advance or too late following the release of software, it may be of little use; the users may forget what they are taught or have already learned the basics by the time training occurs.

Release management is a bridge from project-oriented software development and application selection to production operations. Although testing can be a well-defined and well-executed part of the development life cycle, release management still maintains a level of testing and verification responsibilities. In addition, the core operations of planning, software distribution, and communications constitute the bulk of what is generally considered release management.

Summary

Service delivery depends on a mosaic of interdependent processes, including incident management, problem management, configuration management, change management, and release management. These processes constitute core operations within the SOM model.

The focus here, as with other SOM elements, is to define management tasks in terms of generic operations that apply to a wide range of assets and can be adapted to new technologies as they emerge. The center of management is not desktops, servers, and network hardware but the operations that deploy, maintain, and secure them.

This chapter has introduced the first part of systems management services. Chapter 6 will continue the discussion by examining management issues in service delivery, including service level management, financial management of IT resources, capacity management, and availability management.

Download Additional eBooks from Realtime Nexus!

Realtime Nexus—The Digital Library provides world-class expert resources that IT professionals depend on to learn about the newest technologies. If you found this eBook to be informative, we encourage you to download more of our industry-leading technology eBooks and video guides at Realtime Nexus. Please visit <http://nexus.realtimepublishers.com>.