

realtimepublishers.com[®]

The Definitive Guide[™] To

Enterprise Change Management



Dan Sullivan

Introduction

By Sean Daily, Series Editor

Welcome to *The Definitive Guide to Enterprise Change Management!*

The book you are about to read represents an entirely new modality of book publishing and a major first in the publishing industry. The founding concept behind Realtimepublishers.com is the idea of providing readers with high-quality books about today's most critical IT topics—at no cost to the reader. Although this may sound like a somewhat impossible feat to achieve, it is made possible through the vision and generosity of corporate sponsors such as Merant, who agree to bear the book's production expenses and host the book on its Web site for the benefit of its Web site visitors.

It should be pointed out that the free nature of these books does not in any way diminish their quality. Without reservation, I can tell you that this book is the equivalent of any similar printed book you might find at your local bookstore (with the notable exception that it won't cost you \$30 to \$80). In addition to the free nature of the books, this publishing model provides other significant benefits. For example, the electronic nature of this eBook makes events such as chapter updates and additions, or the release of a new edition of the book possible to achieve in a far shorter timeframe than is possible with printed books. Because we publish our titles in “real-time”—that is, as chapters are written or revised by the author—you benefit from receiving the information immediately rather than having to wait months or years to receive a complete product.

Finally, I'd like to note that although it is true that the sponsor's Web site is the exclusive online location of the book, this book is by no means a paid advertisement. Realtimepublishers is an independent publishing company and maintains, by written agreement with the sponsor, 100% editorial control over the content of our titles. However, by hosting this information, Merant has set itself apart from its competitors by providing real value to its customers and transforming its site into a true technical resource library—not just a place to learn about its company and products. It is my opinion that this system of content delivery is not only of immeasurable value to readers, but represents the future of book publishing.

As series editor, it is my *raison d'être* to locate and work only with the industry's leading authors and editors, and publish books that help IT personnel, IT managers, and users to do their everyday jobs. To that end, I encourage and welcome your feedback on this or any other book in the Realtimepublishers.com series. If you would like to submit a comment, question, or suggestion, please do so by sending an email to feedback@realtimepublishers.com, leaving feedback on our Web site at www.realtimepublishers.com, or calling us at (707) 539-5280.

Thanks for reading, and enjoy!

Sean Daily

Series Editor

Introduction.....	i
Chapter 1: Understanding the Need for Enterprise Change Management.....	1
Rippling Effects of Change.....	1
Making a Simple Change.....	1
Considering a More Complex Change.....	2
Implementing Enterprise-Level Changes	3
Coping with Unintended Consequences	5
The Development of the Need for ECM.....	5
The Rise of Enterprise Applications	6
Integrating with Point-to-Point Solutions	7
Using Integration Middleware	8
Supporting Business Processes	9
Understanding the Impact of Enterprise Applications.....	9
Executive Information Systems Evolve into Data Warehousing.....	10
Meeting Demands of E-Commerce	12
Maintaining a Secure Environment	13
Failing to Meet Expectations	14
Where’s the ROI?	15
Failing Projects	16
Applying Stricter Business Controls to IT Processes	17
Improving Financial Controls.....	17
Managing Risk.....	18
Running an Efficient IT Shop.....	18
Outsourcing Development	18
Improving Operational Integration and Regulatory Demands for Change Management..	19
Leveraging Proprietary Processes.....	19
Mandated Change Control	20
IT’s Growing Responsibilities	22
Addressing Technological Changes.....	22
Adopting Organizational Change	23
Increasing Financial Responsibilities	23
Summary	24
Chapter 2: Examining the Nature of Enterprise Change Management.....	25

Modeling Enterprise Change	25
Constructing a Model of Enterprise Change.....	27
Assets	28
Dependencies	29
Policies.....	30
Roles	31
Workflows.....	31
Using the Reference Model to Define ECM Applications	32
Functional Characteristics of ECM Systems	33
Communicating About Change.....	33
Tracking Changes with AMRs.....	33
Implementation Characteristics of ECM Systems	36
Synchronizing Process and Data.....	36
Using a Shared Metadata Repository.....	37
Sharing the User Interface	37
The Enterprise Change Lifecycle.....	38
Specializing Change-Management Operations.....	39
Identifying Software Configuration Management Requirements.....	39
Identifying System Configuration Issues.....	41
Identifying Content and Document Management Issues	41
Supporting Enterprise Operations with Change-Management Services.....	44
Summary	45
Chapter 3: Managing Change in the Software Development Lifecycle	46
Levels of Change Management in Software Development	46
Understanding the Software Development Lifecycle.....	46
Analyzing Requirements.....	47
Change Management in the Requirements Analysis Process.....	48
Designing Software.....	48
Developing Software	49
Implementing Software.....	50
Maintaining Software.....	50
Retiring Software Applications.....	52
Choosing a Methodology for Software Development	53

Waterfall Methodology	54
Advantages of the Waterfall Methodology	54
Disadvantages of the Waterfall Methodology	55
Spiral Methodology	55
Advantages of the Spiral Methodology	56
Disadvantages of the Spiral Methodology	56
RAD	57
Advantages of the RAD Methodology	57
Disadvantages of the RAD Methodology	57
Extreme Programming	58
Advantages of the Extreme Programming Methodology	58
Disadvantages of the Extreme Programming Methodology	58
Methodologies and ECM	58
The Need for Traceability	59
Demands for Traceability Beyond Software Development	60
Supporting Software Change Management Across the Organization	61
The SW-CMM	62
The SW-CMM and ECM.....	63
Managing Change Within Software Development Projects	63
Using SCM Patterns.....	65
Developing Policies and Processes for SCM.....	66
Version Control Policies	66
Access Control Policies	67
Build, Baseline, and Release Policies	67
Additional Policies.....	67
Summary	67
Chapter 4: Managing Change in System Configurations	68
Interdependence in System Configurations	68
Understanding Implementation Details	69
How SANs Fit in the Infrastructure	69
Details Reveal Dependencies.....	70
Hardware Dependencies	70
Software Dependencies.....	71

Rippling Effects of Misconfigured Systems	72
Defining the Goals of System Configuration Management.....	73
Accounting for System Services	74
Charging for Services	74
Software License Compliance	75
Planning Upgrades and Retiring Equipment.....	75
Redeploying Assets.....	75
Managing Leases, Warranties, and Other Contracts.....	76
Monitoring System Performance	76
Measuring System Load	77
Measuring Resource Usage.....	78
Maintaining a Secure Infrastructure.....	78
Keeping Systems Updated with Patches.....	79
Monitoring Suspicious Activity	80
Logging Vulnerabilities, Breaches, and Responses	81
Fault Management	81
Identifying Failed Components.....	82
Isolating the Effects of Failures	83
Correcting Failure	84
Managing Configurations	84
Modeling System Configuration Management.....	85
Defining System Configuration Assets.....	86
Identifying Dependencies Between Assets.....	86
Developing, Publishing, and Enforcing Policies	87
Roles and Workflows in System Configuration Management	87
Understanding the Unique Challenges of System Configuration Management	88
Summary	88
Chapter 5: Managing Change in Enterprise Content	89
Understanding the Nature of Content in Today’s Enterprise.....	90
The Origin and Proliferation of Content.....	90
Managing Growing Volumes of Content and Multiple Content Systems	90
Managing Unstructured Content.....	91
Managing Distributed Documents	93

When Document Management Is About More than Documents.....	94
Growing Demands of Regulation	94
Managing External Content and Publishers Rights	94
Managing the Content Lifecycle.....	95
Creating Content	96
Approving Content.....	96
Publishing Content.....	97
Revising Content.....	97
Controlling Versions.....	97
Concurrent Editing.....	98
Auditing Changes.....	99
Repurposing Content	99
Writing Logical Units	99
Using Standardized XML Tags	100
Storing Content in Accessible Repositories.....	100
Search and Navigate	100
Archive and Destroy	102
Elements of Change Management for Content.....	103
Assets	103
Managing Changes to Documents	103
Using Appropriate Document Metadata	104
Tracking XML Schemas	106
Formatting and Rendering Specifications.....	106
Identifying Dependencies in Enterprise Content	106
Infrastructure Dependencies	107
Related Asset Dependencies	107
External Dependencies.....	108
Developing Content-Management Policies	109
Staying in Compliance with Policies	109
Maintaining Consistent Document Structures and Metadata.....	109
Controlling Access and Protecting Digital Rights	110
Specifying Roles and Workflows	110
Summary	110

Chapter 6: Practicing ECM.....	111
Varieties of Asset Change.....	111
Simple Change-Management Patterns.....	111
Complex Change-Management Patterns.....	112
Selecting ECM Tools.....	113
Assessing the Functional Requirements of an Organization	113
Determining Asset Types.....	114
Assessing Dependencies	114
Understanding Workflows Within an Enterprise.....	115
Evaluating Access Control and Security Requirements	115
Delivering Information	116
Understanding Technical Requirements and Options.....	118
Work with Existing Platforms	118
Scaling to Perform	119
Integrating ECM Components	121
Administering ECM Applications	122
Estimating ROI of ECM Systems.....	123
Assessing the ECM Market	124
Approaching ECM from a Software Perspective.....	125
Approaching ECM from a Document-Management Perspective	125
Approaching ECM from a Systems-Configuration Perspective.....	125
Understanding the Current State of the Market	126
Introducing ECM to an Organization	126
Assessing the Readiness for ECM	126
Are Change Management Practices Already in Place?.....	126
Is There Executive Sponsorship for ECM?	127
What Problems Are Solved by ECM?	127
Are Daily Operations Constrained as the Result of a Lack of ECM?	128
Deploying ECM.....	128
Benefits of ECM	129
Aligning IT with Business	129
Improving IT Quality Control.....	129
Improving ROI on Development	129

Improving Infrastructure Management	129
Predicting Project Life Cycles	129
The Future of ECM.....	130
Evolving Objects of ECM.....	130
Dynamics of Web Services.....	130
Composite Objects and Logical Records Management	130
Evolving Processes of ECM	131
The Evolving ECM Toolset	131

Copyright Statement

© 2004 Realtimerepublishers.com, Inc. All rights reserved. This site contains materials that have been created, developed, or commissioned by, and published with the permission of, Realtimerepublishers.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 Realtimerepublishers.com, Inc or its web site sponsors. In no event shall Realtimerepublishers.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.

Realtimerepublishers.com and the Realtimerepublishers 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 Realtimerepublishers.com, please contact us via e-mail at info@realtimerepublishers.com.

Chapter 1: Understanding the Need for Enterprise Change Management

Imagine spending a day as an air traffic controller. When you arrive at work, you are confronted with a host of problems: planes arrive ahead of schedule with no available gates, mechanical problems delay the take off of other planes, pilots request changes to flight plans, several planes that are in holding patterns are requesting landing permission (you're not sure why they are in holding patterns, they were that way when you arrived), and ground personnel clamor about missed connections. Sounds rough. Now imagine that on top of this confusion, planes are changing major components in flight, pilots and ground crews speak different languages (translators are few and far between), connections between flights are coordinated by pilots with no centralized control, and the controllers know any change to these operations can create a set of new, unanticipated problems. Now suppose that we're talking about information technology (IT) rather than the airline industry—welcome to a day in the life of an IT administrator.

Change is constant in business, and IT is a nexus for much of that change. Threats and opportunities arise as a result of advances in technology and globalization, changes in government regulation, and a range of other market factors. The challenge facing organizations is not to prevent or even slow change, but to manage it effectively. Doing so is the purpose of enterprise change management (ECM). As I will describe in this chapter, changes in one part of an organization can quickly produce changes, side effects, and unintended consequences in other parts of the enterprise. Recent history is witness to several factors that have introduced and accelerated the pace of change in business—and created the need for ECM.

Rippling Effects of Change

Although the motivation to make a change in an application or process may come from a single source, the effects of that change often spread to multiple points in an organization. A change to A requires a change to B, which in turn, requires a change to C, and so on. In some cases, the ripple effects of a change are localized to a single application or process. More often, the effects extend further as the following examples describe. To effectively manage these series of changes, organizations require ECM systems that capture asset configuration, dependencies, and attributes and provide real-time access to administrators and managers across the enterprise.

Making a Simple Change

A simple change to a database is easily understood. If the longest name allowed in a database record changes from 20 characters to 30 characters, several parts of the database will need to be modified (including the table that stores the name, data entry forms that reference the name, reports that print the name, and extraction processes that move the name to other applications). A single modification creates the need for a series of changes, as Figure 1.1 illustrates. In addition to keeping up with the parts of the database that will require change, IT staff would need to inform users that a new maximum length is allowed and determine whether applications that import data from the database need updating. Administrators still need to confirm that small changes conform to requirements, are tested, and comply with relevant standards. Even the simplest example is not so simple.

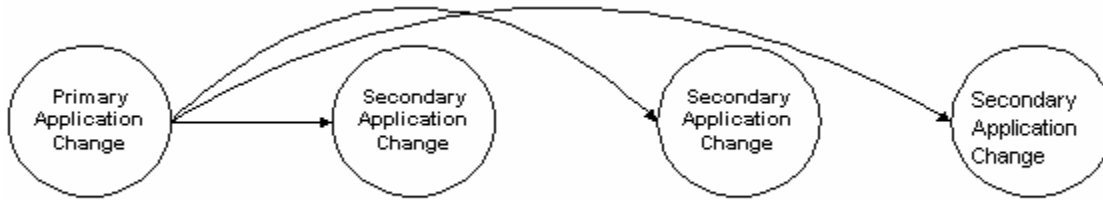


Figure 1.1: A simple change to an application can still lead to secondary changes.

Considering a More Complex Change

Figure 1.2 illustrates a slightly more complex change—a change in one type of asset, such as an application, causes changes in other types of assets, such as business processes, training methods, and documentation. For example, suppose a user upgrades a word processing program. The new application has a variety of new features that require more memory than is available on the user’s desktop computer. The user then places a memory upgrade request, which is handled by tech support. Tech support approves the request, updates the configuration database, and installs the added memory. IT informs the user that despite the installation of the latest version of the word processor, the user will have to save files in a format supported by an earlier version of the program because most desktops in the organization are not running the latest release. Users’ collaboration processes change to accommodate the software upgrade.

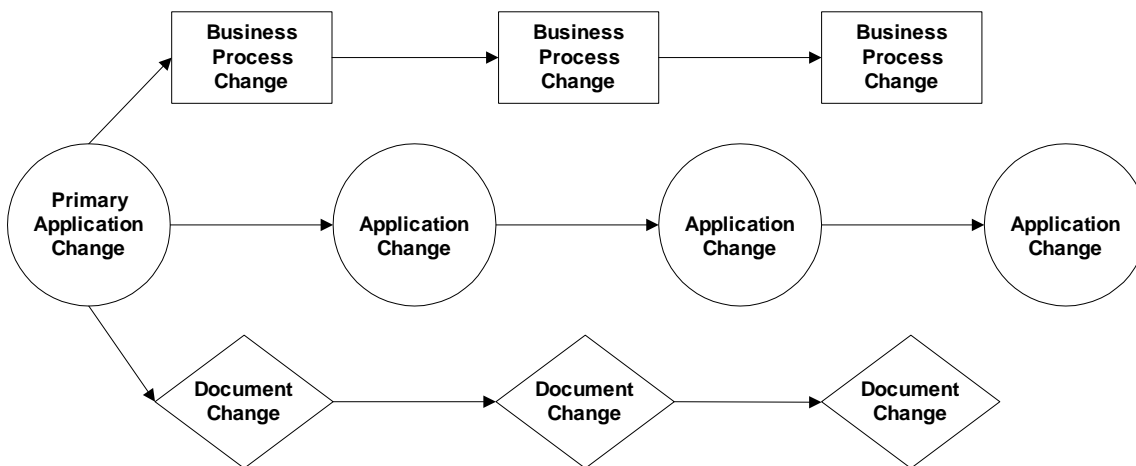


Figure 1.2: Moderately complex changes cause ripple-effect changes to a variety of assets.

Implementing Enterprise-Level Changes

The most complex changes cause secondary changes in a variety of assets, to varying degrees of importance, and in a number of process directions, as Figure 1.3 depicts. Consider the following scenario. An online retailer has received a growing number of complaints from customers about the quality of some of its products. Executives decide that it is time to revise the company's procedures for handling customer complaints. Call center supervisors will now have the authority to override the return material authorization (RMA) system to allow returns after the standard 30-day return period. What needs to change, the executives ask? Off the top of her head, the systems manager cites several changes:

- The RMA system will need the ability to authenticate the supervisor and log a description of the override.
- Programmers will have to change the accounts receivable system, which currently rejects credits for returns after 30 days.
- Managers will need a new report to monitor the use of the new override authority.
- New codes will be added to the RMA system to correctly categorize the new return type.
- The RMA system will need to be modified to allow only supervisors to enter the new return type.

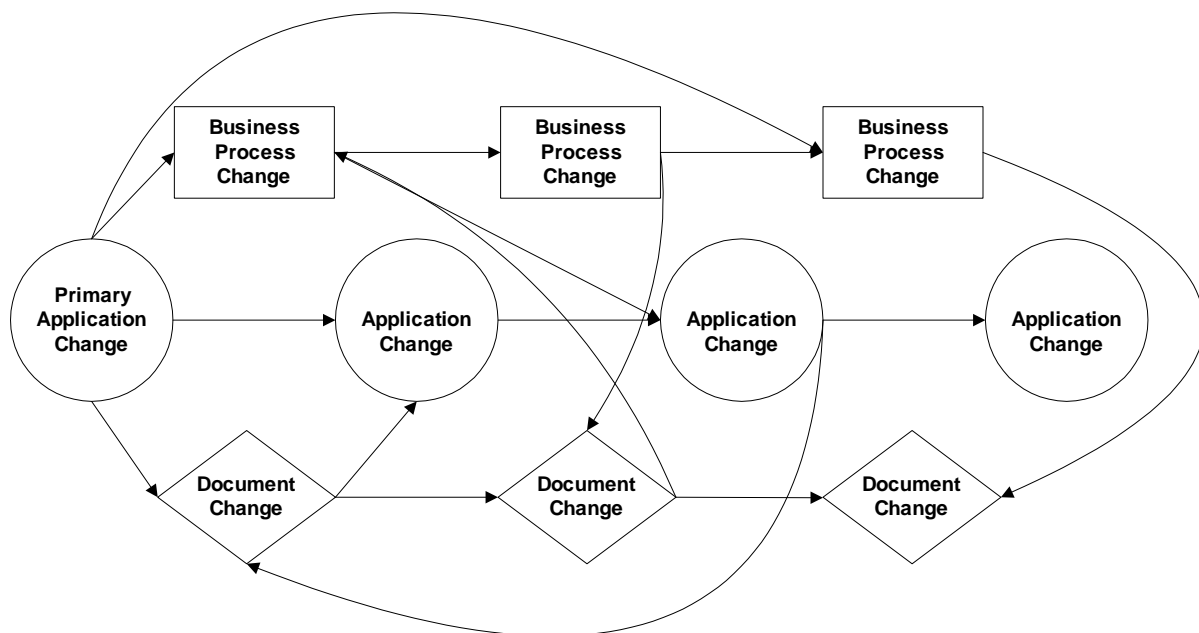



Figure 1.3: Complex changes cause ripple effects in multiple directions and between artifacts in unpredictable patterns.

In addition, the executives realize that the following further changes are required:


- Policies and procedures will need to be updated—executives will need to define guidelines for the use of the override authority.
- Managers must determine how to incorporate email and other customer-contact channels into this process.
- Managers must assess the impact of this change on supply change processes.
- Supervisors and customer support representatives will require training on the new procedures. Performance reviews of supervisors will now include measures of the use of the override authority to prevent abuse.

In addition to these changes, the original change initiates several other business processes, which themselves entail change management. For example, within the IT department, the requested modifications will be evaluated by and approved by the change-control committee, scheduled by an IT manager, developed by a programmer on a development system, tested on a test platform, and deployed to production during a maintenance cycle. Within each of these steps there are procedures for creating a new version of RMA system modules, checking-in and checking-out the code during development, updating system documentation through version control procedures, and so on.

 Changes are rarely localized to a single application or process; however, identifying all related secondary changes is difficult, especially when dealing with distributed systems.

This example highlights several issues common with application and system changes:


- Changes in one process, in this case the return process, cause changes in other processes, such as customer billing.
- Business rules are often distributed and embedded directly in applications. In this example, both the RMA system and the accounts receivable system were hard coded to reject returns after 30 days. Tracking the dependencies and duplication of business logic is difficult without a change-management application.
- One change can initiate multiple change-management procedures. IT has procedures for accepting change requests. The call center has procedures for updating policies and guidelines as well as training supervisors and customer support representatives. Line-of-business (LOB) managers change how they monitor operations using new performance indicators.

 When assessing the scope of changes, be sure to consider how changes in one area, for example IT applications, create the need for change in other areas, such as training and Help desk support.

This example, while realistic, is somewhat idealized. We often do not know how the effects of a change in one system will affect another. How did the systems administrator know that the accounts receivable system would also need to be changed? Are there other systems affected by this change? Changes can easily introduce unintended consequences.

Coping with Unintended Consequences

Suppose that the word processor upgrade described earlier also included new macro features that allow macros to run automatically when a document is open. As a further benefit, macros can now invoke other applications. A user could, for example, open a word processor document and then automatically open a related spreadsheet using this new feature. Several weeks after the upgrade, the user receives an email with a word processor document attached. The document embeds a macro that launches a virus that copies itself to other desktop computers on the network. The virus lies dormant for a few months, then replaces critical operating system (OS) files with bogus versions. Users across the department start to experience unexplained system crashes. The Help desk and IT support respond. They determine the cause of the problem, repair damaged systems, and upgrade virus scanners on desktops. Systems managers configure additional security measures on the email server and train users about the potential problems of macro-based viruses. Finally, the IT department updates policies and procedures to monitor for the newly discovered security threat.

 To avoid such a scenario, when assessing the impact of change, consider the security consequences as well as the functional, documentation, policy, and training changes.

Even when change is managed, the risk of introducing an unintended consequence is always present, especially with the pace of change in technology trends, standards, best practices, and environments. Effective change-management procedures, however, give managers the tools they need to identify the causes of those unintended outcomes.

There is no single cause for the increasing complexity and challenge of enterprise change. Some of the most prevalent sources of change are rooted in trends that reach back a decade.

The Development of the Need for ECM

The business dynamics of the past decade have created unprecedented demands on IT. Deregulation, globalization, technological advances, and evolving management structures apply competitive pressures to every dimension of an organization—research, procurement, production, marketing, sales, and others. These competitive pressures force organizations to operate more efficiently. One common source of inefficiency is poor integration of business processes.

Department-level applications can efficiently address narrow operations. Delivering goods and services, though, typically requires linking a series of these operations. When applications are integrated, the links are fast and cost-effective. When links are based on manual intervention, processes are slowed and costs increase. As Figure 1.4 illustrates, much of the total time required to complete a process can be spent on inter-application manual procedures.

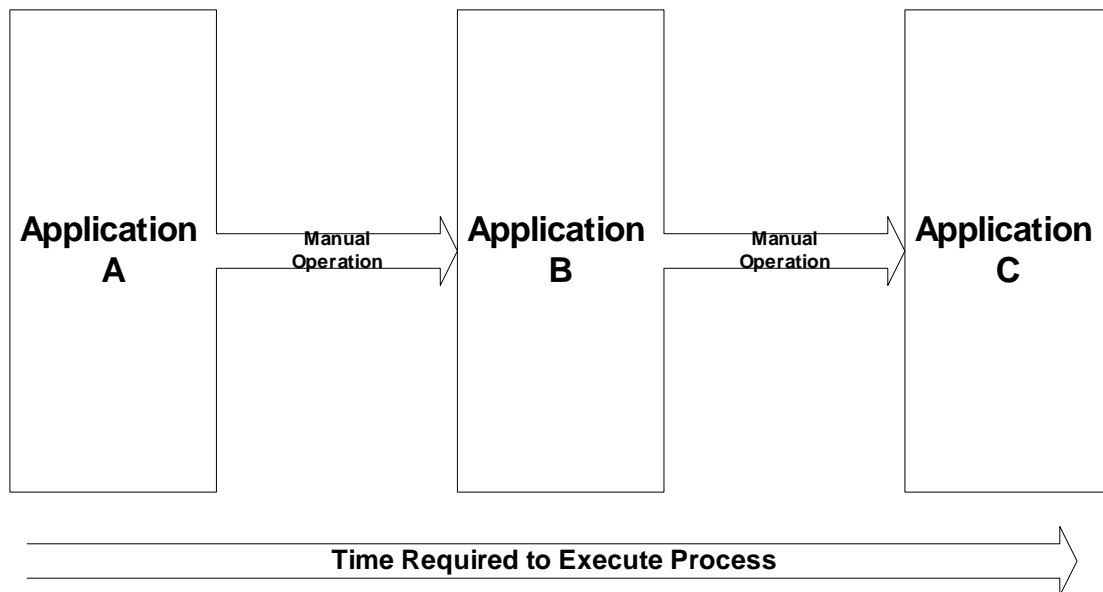


Figure 1.4: When applications are not efficiently integrated, manual operations are often required to complete a business process.

We now understand that additional efficiencies are gained by coordinating and managing change across the enterprise. Designers engineer better products with feedback from the production line. Inventories are better controlled when sales projections drive production schedules. Sales campaigns are more effective when tied to marketing initiatives. These efficiencies are realized when the enterprise uses closed-loop communications across business processes.

Efforts to improve operational integration and business process design were popular in the 1990s. IT has played a central role in several types of strategic initiatives undertaken since then:


- Enterprise applications such as enterprise resource planning (ERP), customer relationship management (CRM), and supply chain management (SCM)
- Data warehousing and advanced analytics
- E-commerce and emerging Web technologies
- Enterprise security and continuity of service

Each of these developments represents a response to the evolving market dynamics that organizations have encountered in the past decade.

The Rise of Enterprise Applications

Enterprise applications such as ERP, CRM, and SCM systems have radically changed how organizations build and deploy applications. Prior to the 1990s, businesses often purchased off-the-shelf applications or built custom systems to meet departmental needs. If the procurement office needed a database to track suppliers and contracts, that department purchased the application. When shift supervisors needed a program to manage workers' schedules, one was built. The shipping department needed a database to track orders, so it put a database in place. The finance department had to track accounts payable, accounts receivable, and payroll, so it used an accounting package. These solutions typically met the localized needs of each department; unfortunately, these systems rarely worked well together.

In many organizations, the various applications often duplicated data. Customer information could be found in the shipping department's database as well as through the accounting package. Employee data was tracked in department scheduling programs as well as through the payroll system. In well-run IT shops, data is shared between systems. The payroll system, for example, might provide source data for the shift scheduling program. One system is designated the master, the others are the targets, and all changes are made in the master and propagated to the target systems. In practice, this scenario solves problems, but creates many problems as well.

 Organizations have long realized that integrating department-level applications improves efficiencies. However, the cost of integration has been a significant barrier to implementing integrated systems.

Integrating with Point-to-Point Solutions

This master-target scenario requires that the systems administrator develop or purchase a program to extract the data from the source system and load it into the target. (This requirement, in turn, creates yet another application to maintain.) The organization now requires procedures to control when the data loads occur, describe how to recover from failures, identify who is responsible for data and application maintenance, and expound on other change-management issues. However, the most troubling aspect of this approach is the potential for a growing number of systems needed to support these point-to-point integrations.

As noted earlier, organizations gain efficiencies by integrating applications that support a business process. In the past decade, IT staff has found itself linking applications in varying combinations because many applications were designed with departmental requirements, rather than business process requirements, in mind (see Figure 1.5).

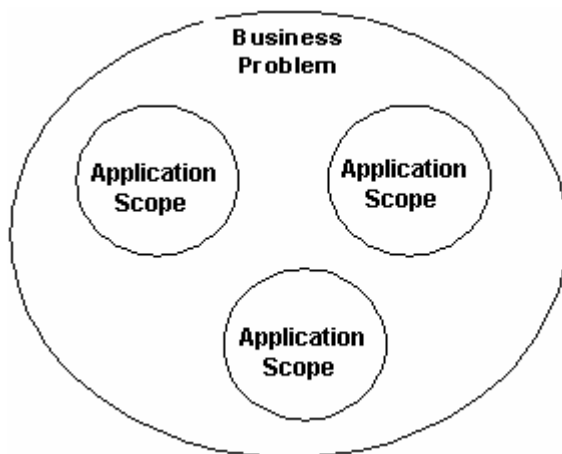



Figure 1.5: Narrowly focused applications rarely support a complete business process.

Developers are left with the often tedious, error-prone job of linking these separate systems. Batch extracts and loads are a common, but crude, integration technique. In an attempt to improve data-sharing processes and minimize the chance of “reinventing the wheel,” developers have begun using middleware applications to link disparate systems.

 Point-to-point integration is a partial solution to the problems that businesses encounter through a silo-application setup. Unfortunately, this integration introduces new change-management issues. As the number of point-to-point integration systems increases, change management becomes more difficult.

Using Integration Middleware

Middleware is a class of applications that provide a common method for moving data between systems and invoking applications or procedures from external systems. Early middleware systems, such as remote procedure calls (RPCs) and remote database access, offers developers a common transport mechanism for moving data and invoking services. It does not, however, solve the larger organizational problem of aligning IT systems with business processes.

Middleware has evolved into more sophisticated mechanisms such as message queues and distributed transaction processing systems. With these techniques, developers can more easily implement full business operations. For example, when a patient submits a healthcare claim, the insurance company will need to verify the patient's coverage, validate the physician information, calculate deductibles, issue payments, and generate explanation-of-benefits letters. These steps are all part of the claims processes that might require several applications that are governed by complex workflow rules. If one step fails, the entire process fails. It is difficult to implement transaction processing logic using point-to-point integration schemes. Middleware, however, gets us one step closer to the ideal of one unified application for a business process.

Middleware provides a general mechanism for moving information between applications, as Figure 1.6 illustrates. One of the key advantages of middleware over point-to-point systems is the support for distributed transactions. These transactions allow developers to treat an entire business process, such as claims approval, as a logical unit of work. If one of the systems required by the process is down, the claim is not processed, and the operation is rolled back. Applications do not need to be aware of the implementation details of other applications in the process because these details are handled by the middleware.

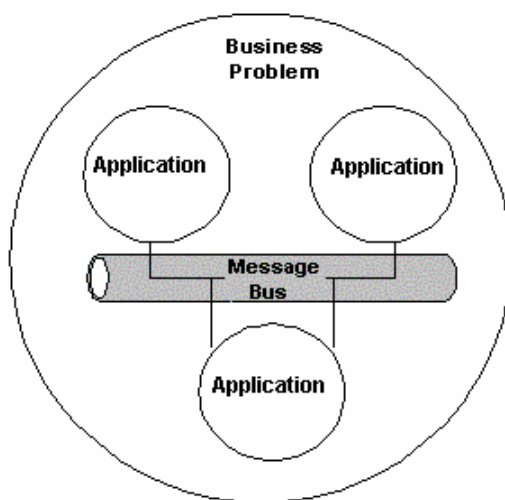


Figure 1.6: Middleware, like message queues, reduce point-to-point connections but still do not solve the fundamental mismatch between applications and business processes.

Middleware solutions are definitely an improvement over point-to-point systems, but middleware solves only part of the problem. Business processes are still dependent upon multiple independent systems, each with their own purpose and lifecycles. Changes in one system can impact another. Dependencies are not always clear or well documented. When business processes change, multiple systems will have to change (and each of those changes potentially creates ripple effects causing still further changes).

Supporting Business Processes

Enterprise applications promised to solve these problems, and they did—to some extent. ERP systems integrate large-scale business processes such as manufacturing operations. Databases are unified and information is shared across departmental boundaries. Point-to-point integration is no longer necessary (or is at least significantly reduced). Middleware, when needed, is integrated into the heart of the ERP system.

However, implementing enterprise applications is more challenging and expensive than many adopters anticipate. Failures and delays are common. The industry and its customers underestimate the difficulty of implementing these systems. There are technical challenges, of course, but another source of problems centers on change management.

Unlike earlier system implementations that required coordination within a department, ERP, CRM, and SCM applications span organizations. Changes in one department ripple through to affect others. Thus, organizations, including designers and developers, have to change how they work. Programming mavericks ready with quick fixes for requirement changes and bugs are not welcome, and coordination is critical. As the scope of applications expands, so does the impact of changes.



Enterprise applications solve many process integration issues but introduce a new level of change-management considerations.

Understanding the Impact of Enterprise Applications

Enterprise applications change the way employees work. These applications change the way supervisors manage. They change executives' expectations. They change IT. As previously mentioned, three common forms of enterprise applications are:

- ERP—Attempt to integrate production-oriented operations across the enterprise
- CRM—Provide an integrated view of customers
- SCM—Address procurement and vendor-management issues

Each type tackles organizational processes from a different perspective, but all three cross departmental and sometimes LOB boundaries. Another common characteristic among these enterprise application types is that some of the most significant costs of these systems entail change, including:

- Training
- Customization
- Data conversion
- Integration

Just as transaction processing systems evolved into enterprise-scale systems, decision-support systems grew to encompass broader views of the organization. The following list highlights the effects of enterprise applications on change management:

- Amplified the effect of changes—consequences ripple throughout the organization
- Brought tighter integration of IT and LOB units
- Reduced the number of point-to-point integration systems
- Introduced, at least in the short term, data migration projects

Executive Information Systems Evolve into Data Warehousing

Executives and managers have long needed management information about production levels, sales quotas, quality control, and other key performance measures. Prior to the emergence of data warehousing, executive-management and decision-support systems drew data from several systems in an ad hoc manner to provide a unified view of core business processes. These early systems required custom coding and were often planned around immediate requirements rather than specific design principals. Fortunately, change management was easily controlled because these systems had few users and a small number of developers.

In addition to the development of enterprise applications, the 1990s witnessed the rise of a more systematic approach to executive information systems—the data warehouse. The data warehouse was not only a reporting system but also an enterprise-scale application for gathering, storing, and analyzing a broad range of topics. Data warehouses provide a historical, integrated view of an organization.

Data warehouses consist of four principal components: source systems; the extraction, transformation, and load system; the data repository; and reporting and query tools. Figure 1.7 shows a typical configuration.

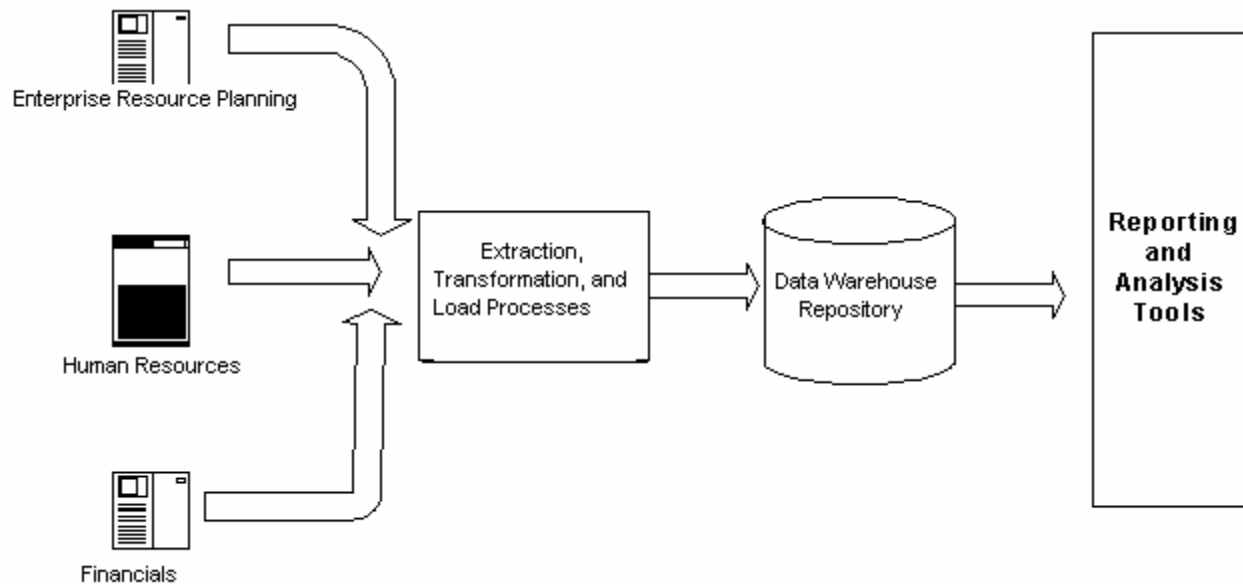


Figure 1.7: Data warehouses provide a framework for standardized delivery of decision support information.

Source systems are typically transaction-oriented applications such as ERP, CRM, and financial-management systems. Multiple sources are typically required to gather all the information required to report on key performance indicators.

Extraction, transformation and load (ETL) systems collect data from the source systems. ETL systems then combine data streams, reformat data, derive calculated measures, and store aggregated data in a format suitable for use with ad hoc query tools.

Data repositories are usually either relational databases (such as Oracle 9i, IBM DB2, and Microsoft SQL Server) or online analytic processing (OLAP) databases. Relational databases are reliable, scale well, and provide robust management services. OLAP databases are designed for rapid response to multidimensional queries such as, what is the difference between this year's sales in the eastern region and last year's sales in the same region?

Reporting and analysis tools provide a range of functionality. Dashboards provide summary information of key performance indicators. Ad hoc query tools allow users to “slice and dice” data along multiple dimensions. OLAP tools support advanced functionality such as time series analysis and forecasting.

Data warehouses have emerged as the standard for delivering business intelligence to users across the enterprise. With this service, come demands on ECM. The following list highlights the effects of data warehousing on change management:

- Introduces new applications and procedures, such as
 - New databases and sometimes new *types* of databases, such as OLAP systems
 - Extraction, transformation, and load processes and their related systems
 - Reporting and analysis tools
- Introduces new users to analytic reporting and analysis
- Changes in source applications impact the data warehouse repository, ETL processes, and, in some cases, end-user reporting

Meeting Demands of E-Commerce

When e-commerce emerged in the late 1990s, the hype about the Internet changing the nature of business overshadowed the real changes underway in many organizations. (The fundamentals of business did not change—well-run companies that meet customer expectations and generate shareholder value survive, others do not.) The real story of change centered on how a new sales channel, the Web, provided low-cost access to a vast market while simultaneously opening the organization to a wide array of competitors. At the same time, the Web brought business closer to suppliers and enabled more efficient procurement operations.

Initially, businesses needed to implement basic applications, such as online catalogs, content-management systems, and secure transaction processing. Then they needed to address customer support, marketing, promotion, and other ancillary functions. The most innovative businesses integrated the Internet channel directly into ERP and CRM systems. Multi-channel selling—for example, call center support for online shoppers—further improved customer options while introducing more complexity to the IT infrastructure.

E-commerce makes demands on virtually every area of IT. Mainframe and client/server programmers were finding that their programs needed to adapt to a new development model. Network administrators and engineers became well versed in firewalls, router security, demilitarized zone (DMZ) architectures, key systems, and virtual private networks (VPNs). Operational managers accustomed to nightly maintenance windows had to adjust to 24 × 7 system uptime. But IT was not the only department affected.

Selling online is not the same as selling in the brick-and-mortar world. The Web lets customers move to a competitor's site in a few clicks of a mouse. Marketing executives, copywriters, and graphic designers had to change their craft to conform to the Web audience's expectations. Usability became a focus of attention. Technical staffs started working directly with their marketing colleagues. The Web was eliminating degrees of separation within businesses. The following list highlights the effects of e-commerce on ECM:

- Introduces a new sales channel that has distinct marketing and sales requirements
- Enables more efficient procurement and supply chain management procedures
- Increases complexity of network architecture and management

- Introduces a new development platform to IT
- Requires IT to work more closely with more “business-oriented” departments
- Increases visibility of errors and poor communications/processes
- Increases the cost of errors and the impact on business/revenue

Maintaining a Secure Environment

The Internet brings threats as well as benefits to organizations. Unless properly configured, networks open businesses to a range of threats from system disruptions to theft of confidential information. Organizations are subject to:

- Viruses—transmitted through emails—that destroy files or disrupt email systems
- Denial of service (DoS) attacks that overwhelm servers with bogus service requests
- Malicious programs that lie dormant until invoked to damage their host or attack other servers
- Theft of proprietary information
- Increased security risks as a result of outsourcing and collaborating with partners and suppliers

Reducing security threats requires detailed knowledge of system vulnerabilities and methodical attention to change management. As new systems are introduced to networks, the systems must be secured. IT staff must configure local elements, such as administrator passwords, system processes, and network ports to provide required services without running unnecessary processes that might be vulnerable to attack. Systems administrators work to ensure that new servers don't operate in conjunction with other systems to open the network to hackers. For example, a hacker may not be able to connect directly to a vulnerable database server but could tunnel through HTTP and, once inside the firewall, execute an attack on the database.

Systems administrators today maintain a guarded skepticism when introducing new applications. As many have realized, sometimes too late to prevent problems, the introduction of new technology entails risks. As researcher Diomidis Spinellis noted in *Software Reliability: Modern Challenges* “[M]odern computer processors are often delivered with errors....Operating systems and programming languages are becoming increasingly complicated and their implementations less trustworthy. In addition, component-based multi-tier software system architectures exponentially increase the number of failure modes, while Internet connectivity exposes systems to malicious attacks. Finally, IT outsourcing and blind reliance on standards can provide developers with a false sense of security.”

 You can find this paper at <http://www.spinellis.gr/pubs/conf/1999-ESREL-SoftRel/html/chal.pdf>.

Again, as with enterprise applications, data warehouses, and e-commerce, a series of fairly localized changes in the overall IT infrastructure creates a web of interdependencies. The following list offers the effects of security threats on ECM:

- IT network and systems administrators must monitor systems for a variety of threats.
- IT staff must install OS, network, and application patches as needed.
- Users must be trained about how to control threats from email and Web use.
- Configuration changes potentially introduce new vulnerabilities.
- Applications must be certified and validated as compliant.

Failing to Meet Expectations

As we've explored in the last section, the IT administration profession has adapted rapidly changing technologies to evolving business structures and processes. From implementing enterprise applications, business intelligence, and e-commerce systems to responding to security threats and protecting computing infrastructure, IT departments have played a central role in the success of many organizations. But they have also been closely tied to failures.

For IT professionals, it is difficult to hear that many of our projects have been failures; that if cars were designed like computers, they would crash twice a day; or that projects will not be funded because they do not generate a sufficient return on investment (ROI). The perception that IT has failed to meet many expectations is fueled by three distinct problems:

- Insufficient ROI for many projects
- Unreliable and vulnerable systems
- High rates of failure and partial success in delivery business systems

IT has had more successes than failures, and it remains an essential element in business strategy for virtually all organizations, but that does not get IT off the hook. As change-management consultant Jim Love described it in "Return to Value" (*Intelligence Enterprise*, May 13, 2000), "Every period of excess has its hangover. The last decade ended with the revenge of the nerds, but this one begins with the revenge of the accountants. The free-flowing cash is gone, and resource-strapped companies have found themselves struggling to make every dollar count. No corporate division has paid a greater price for past excesses than IT."

 Read "Return to Value" at http://www.intelligententerprise.com/030513/608feat2_1.shtml.

Where's the ROI?


American companies started the 1990s allocating 30 percent of capital expenditures to IT; they ended the decade spending almost 50 percent of all long-term investment on IT (as cited by Nicholas Carr in "IT Doesn't Matter" in the May 2003 issue of *Harvard Business Review*). Unfortunately, ROI was one acronym rarely heard in developer bullpens, machine rooms, and IT conference halls. A combination of intuition, herd mentality, and blind faith in technology allowed IT to avoid the rigorous capital expenditure analysis to which other business units were subjected. Those days are gone. IT projects have to pass the same financial muster their brick-and-mortar counterparts do.

To pass muster, ROI analysis must be based on quantifiable, tangible savings. Are you deploying a supply chain-management portal to reduce the number of documents mailed to vendors? Doing so leads to quantifiable savings in printing and postage and is likely a sound investment. Are you deploying an enterprise information portal so that knowledge workers can collaborate and save time searching for information? The ROI for this project is not as clear as for the previous example, but the project may still be justifiable. Do you want to deploy an HR portal so that employees can get corporate news, look up leave-time balances, and check email from the Web? As far as ROI is concerned, this project doesn't stand a chance. A combination of poor performance in high-profile projects and increased visibility of IT projects is driving the demand for better control of IT capital spending.

 The increasing visibility of IT projects contributes to the need for effective ECM.

Trade magazines regularly feature stories of troubled big-ticket projects at Fortune 500s, overspending on software licenses, and "how-to" techniques for saving failing implementations. For example, in 1999, Hershey made news when its broad SAP/R3 implementation ran into trouble during one of its busiest times of the year. However, internal concerns are not the only factors driving IT to better manage capital projects.

CEOs and CFOs are always pressured by market conditions, but changes in government regulation are adding momentum for them to better understand IT expenditures. In the United States, the Sarbanes-Oxley Act of 2002 enacted widespread changes in corporate governance and financial disclosure. CFOs are signing off on quarterly financial reports that include IT expenditures, so accurate reporting is essential. Corporate officers, boards of directors, and shareholders are closer now to enterprise projects than ever before.

 Government regulation as well as market factors are accelerating the demand for ECM.

Failing Projects


As if poor ROI isn't bad enough, businesses have received systems that did not meet requirements or reflected inaccurate requirements. Consider the following:

- One global enterprise spent \$10 million on a CRM system after consulting with staff and signing off on functional requirements. When the applications was deployed, front-line workers rejected the system saying the “interface and workflows were cumbersome, confusing, and unfriendly.” (The source for this information is Terry Jabali’s “CRM: Key Imperatives for Success” available at http://www.intelligentcrm.com/feature/2003/05/0305feat1_1.shtml.)
- According to the Standish Group, 23 percent of IT projects failed in 2000, down from a 31 percent failure rate in 1994.
- As cited in Demir Barlas’ “A.T. Kearny Rates E-Procurement” (*Line56 Media*, June 29, 2002), supply change-management systems are leading to more than 40 percent improvement in cycle time, a 10 percent reduction in staff, and a 13-to-1 ROI when effectively implemented; however, only 8 percent of companies implementing e-procurement are realizing those returns.

Clearly, tighter communication between users and designers and architects is required. More emphasis is needed on processes and workflows rather than simply on isolated behavior of individual components and subsystems. Unfortunately, if communications are clear and development is managed, the ultimate results might still be less than expected.

Meeting the localized needs of a department or LOB does not guarantee the best outcome for the enterprise as a whole. “[I]t has been demonstrated time and again that local optimization leads to global suboptimization, contrary to the popular belief that what is good for the function is good for the enterprise. As an example, a functional area might “batch” inbound work so that each item dealt with the least common effort, although that ultimately delays almost every aspect of the process” (quoted from Alec Sharp’s “A Briefing on Business Process” in *Business Briefing: Data Management and Storage Technology 2002* at http://www.wmrc.com/businessbriefing/pdf/data_2002/publication/sharp.pdf).

Increasing spending on IT rarely translates into superior financial results for companies overall. In the results of a survey performed by Alinean of 7500 large United States companies, the top 25 performers spent only 0.8 percent of their revenue on IT while the average company spent 3.7 percent. A similar study by Forrester Research showed those companies that spent the most on IT rarely realized the best results (cited by Nicholas Carr in “IT Doesn’t Matter”). If Carr is correct, and the numbers seem to back his argument, then managing risk, using technology efficiently for generic operations, and leveraging innovative, proprietary processes are key to aligning business and IT strategy.

 Beware of measuring local impact of change, it may not correspond to the overall benefit or cost to the organization.

Applying Stricter Business Controls to IT Processes

Businesses in general and IT departments in particular are reacting to the failures, missed opportunities, and changing market drivers of the recent past. Some of these changes implement improved financial controls, some create closer collaboration between IT and LOBs, and others draw IT deeper and broader into business processes that span the enterprise.

Improving Financial Controls

IT operations are becoming more closely aligned with LOBs. This realignment comes in several forms:

- Moving IT staff from centralized technology groups into business units that have bottom-line responsibilities
- Sharing responsibility for profit and loss between IT departments and LOBs
- Eliminating IT budgets—funding for projects and operations are charged back to LOBs

Closer alignment of IT operations with business units' day-to-day operations improves communication and control of technology projects. Situations such as the previously cited example of the CRM system that was cumbersome, confusing, and unfriendly are less likely to occur when users “in the trenches” actively participate in system development.

Weaving IT operations into the fabric of LOBs also allows for improved project portfolio management. When IT projects are segregated from other projects, the organization is likely to miss the optimal combination of projects. This missed opportunity can occur in two ways: First, because projects are selected from separate groups, the value of each is compared with others in the group instead of from across the organization. Second, the ultimate set of projects selected may not be the best mix of projects. For example, an e-commerce initiative funded through IT may not realize its potential ROI because a complementary marketing campaign was not funded. Techniques such as enterprise portfolio analysis prevent this type of problem by determining the optimal mix of projects, investments, and activities based upon an organization's strategic goals. The trend toward value management and portfolio management can lead to a cost savings of more than 30 percent according to some industry analysts.

This closer union between business and IT units has led to several successful patterns for improving the overall benefit of IT operations, including:

- Managing risk
- Focusing on IT efficiencies
- Outsourcing development

The relative benefit of each of these approaches varies among organizations, but the following sections describe the most common benefits and risks.

Managing Risk

Risks to business from IT infrastructure have many forms—security breaches, system failures, and unintended consequences of changes, to name just a few. Minimizing risks requires technical knowledge as well as an understanding of business processes. As we've explored, as systems become more integrated, changes to one application can cause ripple effects that propagate to other systems. LOB managers and system designers need detailed information about interdependencies, process flows, and the timing of events. Silo-based change-management procedures are no longer sufficient.

Running an Efficient IT Shop

Efficient use of technology not only requires vision but also requires technical savvy and business discipline. Areas for obvious savings include software licenses. According to one Gartner study, companies that buy more licenses than they actually need will increase total cost of ownership (TCO) of the software by 20 to 30 percent by 2005. Do companies even need to spend any money on software licenses? This question is gaining greater scrutiny as companies investigate the viability of open-source software.


Open-source OSs and Web servers such as Linux and Apache are the most well-known open-source products, but a wide variety of products are available under open-source licensing. Even commercial enterprise applications have open-source competitors such as Ohioedge CRM Server (<http://www.ohioedge.com/factsheet.html>) and Compiere ERP and CRM Business Solution (<http://www.compiere.org/>), both of which are designed for the small and mid-sized company market. Organizations are more likely to consider open-source alternatives as companies realize that the success or failure of a project is not only determined by the software but also by how business processes are implemented and adapted to the applications.

Outsourcing Development

Another trend that demonstrates the commoditization of IT is outsourcing. Companies with core competencies in financial services, healthcare, transportation, energy, and retail are applying the long-understood principal of division of labor to IT. Standardization of hardware, enterprise applications, and networks are reducing barriers to outsourcing. Simultaneously, globalization is creating opportunities for cost-competitive IT service firms outside the United States and Europe to enter lucrative international markets. The trends are clear (the following statistics are from the META Group):

- The overall offshore application outsourcing market, currently valued at \$6 billion, is growing at 20 to 30 percent per year.
- In Europe, the outsourcing market is growing at 18 percent per year.
- By 2005, most European IT organizations will outsource at least one mission-critical technology.

Outsourcing creates new management challenges. Designers and architects may be half a world away from developers implementing designers' plans. Understanding requirements is difficult enough in close-knit teams; international outsourcing adds geographic distance, time delays, and the potential for cultural misunderstanding. Much IT is now a commodity. Strategic advantage is gained not by technology alone, but by the integration and optimization of business processes within the corporation and beyond to suppliers and customers.

 Technology alone does not provide strategic advantage to a company. Competitive advantage arises from applying technology to proprietary business processes.

Improving Operational Integration and Regulatory Demands for Change Management

Executives and LOB managers are realizing the need to manage business processes and related changes across the enterprise. Both market forces and government regulation are driving this trend.

Leveraging Proprietary Processes

Operational integration can lead to significant gains when proprietary business processes are extended by traditional organizational boundaries. For example, consider Dell Computer Corporation and Walmart. Both companies sell commodity products in highly competitive markets, yet they are more successful than most competitors. One technique used by both companies is tight supply-chain integration.

After September 11, 2001, the sale of American flags surged and two retailers, Walmart and Target, sold a majority of the flags. Walmart's point-of-sale system is tightly integrated with its suppliers, so replacements were automatically ordered as inventory began to drop. By the time Target placed its restocking orders, it was too late—the sole supplier was out of stock. Wal-Mart beat Target to the punch.

Dell also uses close-knit supply-chain integration. In addition, the company extends its information sharing to customers. Customers are segmented into eight categories: global enterprise accounts, large companies, mid-sized companies, federal, state and local, education, small companies, and consumers. Programs are customized for each segment based on information gathered from customers. As Michael Dell describes it, information about customers is essential to the company's success. "It's a key part of why rivals have great difficulty competing with Dell. It's not just that we sell direct, it's also our ability to forecast demand—it's both the design of the product and the way the information from the customer flows all the way through manufacturing to our suppliers... We simply couldn't do it without customers who work with us as partners" (quoted from Joan Magretta's "The Power of Integration: An Interview with Dell Computer's Michael Dell" *Harvard Business Review*, March-April 1998). In some cases, tight management controls are not optional, they are required by law.

Mandated Change Control

Change-control boards are centralized bodies that review change requests and project plans, monitor project development, and provide a communication channel for stakeholders involved with application development. The advent of change-control boards brings new policies and procedures that govern IT. Change-control boards may focus exclusively on IT operations; however, in many industries, it is becoming clear that change must be managed across full business processes. Government regulations, for example, are driving several industries to broaden the scope of change control.

Regulatory demands from federal, state, and international governments are imposing strict requirements on change management within particular industries. Several recently enacted laws in the United States, for example, are forcing organizations to implement change-management procedures that span the breadth of operations:


- **Gramm Leach Billey Act (GLBA)**—Targets the financial services industry. The GLBA, also known as the Financial Services Modernization Act, governs financial data that identifies individual consumers. Much of the impact of this law is on privacy. For example, financial institutions must provide consumers with information about the institution’s privacy policies and practices. The businesses may not disclose private information to unaffiliated third parties unless the consumer opts out of the law’s protections. The law further requires banks to complete a study of information sharing practices. This law, in effect, defines boundaries around the type of information sharing with business partners that has proved so beneficial for Walmart and Dell. Financial institutions can still devise innovative programs that involve information sharing, but these programs must be developed in a well-controlled and auditable manner.
- **Health Insurance Portability and Accountability Act (HIPAA)**—Regulates the healthcare industry. HIPAA is a broad-sweeping regulation governing many aspects of the United States’ healthcare system. In effect, the regulations are mandating the development of ECM procedures that govern a broad set of enterprise information. Two HIPAA rules with direct impact on ECM are known as the Security Rule and the Privacy Rule. These directives require:
 - Healthcare organizations to safeguard protected healthcare information (PHI) by assessing organizational needs, defining and implementing protective procedures, and using risk assessment methods to balance protections with the cost of implementing those safeguards.
 - Organizations that gather and store PHI to know when and where personal data moves through the enterprise and how it is shared with business partners.
 - All personally identifying health care information, in any medium (not just electronic), be protected.
 - Healthcare organizations to use role-based access controls to limit access to protected information. They must also create policies and procedures to identify who can access particular types of information and why they need access to it.
 - Business partners to establish agreements to ensure that data is protected even after it leaves control of the original source organization.

- 21 Code of Federal Regulation Part 11—The United States Food and Drug Administration (FDA) has initiated a program to identify current good manufacturing practices (cGMP) for the pharmaceutical industry. The objective of the program is to improve the overall quality of medications while promoting innovation and continuous improvements in the industry. The regulations regarding cGMP, referred to as 21 CFR Part 11, cover the breadth of pharmaceutical manufacturing. Topics addressed include:
 - Organization and personnel
 - Buildings and facilities
 - Equipment
 - Components, containers, and closures
 - Production and process control
 - Packaging and labeling
 - Distribution
 - Laboratory
 - Reports and records

Across these topics, there is an emphasis on the importance of data integrity and records management. These regulations mandate that pharmaceutical companies implement two dimensions of ECM: application change control and data integrity. Once operational systems are validated, change-control procedures must be in place to ensure unapproved changes are not made. According to the FDA, changes made after systems are deployed are the cause of almost 80 percent of all software failures in recalled medical devices. Change-management controls, including audit trails, are key tools for identifying and correcting those bugs.

The regulations also specify record retention requirements for data generated by these systems. Data must be accessible for the duration of the record retention period, so manufacturers have two options for managing the data. The *time capsule* approach uses the original electronic media and system that generated the data. With constant improvement in instruments, software, and storage media, this approach requires that companies maintain legacy systems to remain in compliance while upgrading technology to remain competitive. The second approach, *electronic records migration*, copies records to new storage devices as the devices become available. With this method, companies must ensure the integrity of data throughout the records migration process. This requirement is met through the use of validated systems, audit trails, and enforced policies and procedures. Change-control procedures enable companies to remain in compliance with regulations without unnecessary expense or the need to maintain obsolete systems.

Each of these laws defines requirements for the way in which data is accessed, stored, transmitted, and audited. These are not technology regulations, they are *enterprise-wide regulations* covering several business processes. Change management is no longer the sole responsibility of one department or LOB but spans the organization.

 These three regulations are just examples of the types of regulatory requirements that are affecting a wide range of industries. Other governments, including state governments in the United States and the European Union, are also passing regulations that make greater demands on ECM.

Government regulations are introducing new demands for ECM. Individual applications and business processes are not regulated—broad abstract entities, such as healthcare information, and entire production processes, such as the development of drugs, are the subject of these laws. Silo-based approaches to change management are no longer sufficient to meet these requirements. Change-management processes must encompass the entire organization, address procedures and systems and artifacts, and support closed-loop communication to maintain control of operations.

IT's Growing Responsibilities

IT departments across industries are assuming more responsibilities. Businesses can no more function without IT than they can operate without electricity or telephone service. Technology is woven into the fabric of business processes. The processes span the organization and, by necessity, so does IT's reach. IT's responsibilities are growing along three dimensions:

- Technical
- Organizational
- Financial


Each category represents a distinct set of demands.

Addressing Technological Changes

The demands on IT departments from a technical perspective are well understood. IT staff are responsible for a range of systems, including:

- ERP, CRM, SCM, and other enterprise applications
- Analytic and business intelligence applications, including data warehouses, OLAP systems, and data mining systems
- Enterprise information portals
- Networks and security infrastructure

Each of these areas requires distinct expertise and careful attention to interdependencies between systems. In addition to the breadth of applications, the characteristics of these systems are changing. For example, as Web services become more established, they could radically increase the number of integration points between systems. Businesses are demanding information faster. Real-time data warehousing provides updated information to decision makers throughout the day. Integrated supply-chain-management systems update inventories and initiate orders as goods are purchased. IT permeates most, if not all, core business processes.

 As systems become more integrated and as the need to keep systems constantly online grows, the maintenance challenges increase. ECM techniques are needed to assess the impact of changes. Executives and managers are demanding information that allows them to proactively prepare for change and reduce emergency change orders.

Adopting Organizational Change

From the organizational perspective, IT serves a diverse user base. Internal users range from finance and HR executives who are managing to key performance indicators to scientists and engineers who are using a wide range of instruments and specialized software for research and development. Of course, the user base does not stop at the company doors. Supply-chain partners are integral components of core business processes. Customers are a growing base of users for many organizations. The Web enables direct sales, marketing, and support to customers. And Web self-service is rapidly emerging as a cost-effective alternative to call center support for many businesses.

The variety of needs of these distinct users requires that IT not only maintain systems but also understand how customers' needs relate and create networks of dependencies. For example, if a mobile telephone manufacturer provides online access to technical support databases, resellers can deploy self-service Web support directly to the manufacturer's customers. The reseller must then maintain close communication on changes to the support database that affect end users. Analyzing patterns of use in the self-service site supplied by the reseller can provide insights to the manufacturer about usability, reliability, and the need for new features.

Increasing Financial Responsibilities

In the past, IT departments were considered overhead operations. Their services were not directly related to revenue-generating operations. Decisions about which new technology to deploy or which systems to upgrade were made based upon a broad conception of the overall good of the company. More and more, IT services are tied directly to costs and revenue lines. CIOs are responsible for profitability. IT departments charge back expenses to LOBs that make decisions about technology expenditures. In some cases, IT staff report directly to managers in operational units. This distribution of financial and human resources make management and coordination of IT operations even more difficult than when IT was highly centralized.

IT staffing models are also changing. Outsourcing is increasing and companies are turning to subcontracting to maintain staffing levels without increasing the number of full-time employees. Virtual teams bring vendors, consultants, subcontractors, and internal staff together on an ad hoc basis to meet specific objectives.

At a time when IT responsibilities are growing, the traditional centralized-management structure is being eliminated. This transformation compounds the need to create and maintain closed-loop communication channels so that changes in one area of operations are accommodated by related operations. In effect, IT organizations are asked to manage more complex infrastructures with fewer staff and more complicated management structures. Clearly, new types of procedures and tools are required to control this dynamic environment.

Summary

Although we might sometimes think we are in the middle of a new and radically different world from anything seen before, we should remember Oliver Wendell Holmes' insight that "a page of history is worth a volume of logic." Organizations are managing new competitive pressures, leveraging new technologies, and adapting to shifting customer expectations and government regulations. Those that will survive and thrive will do so by focusing on business fundamentals:

- Maintaining close communication between related LOBs
- Controlling costs
- Integrating business processes
- Proactively managing change across the enterprise

We only need to consider the history of the mainframe to see the persistence of patterns in the history of business and technology. In the early days of IT, mainframes were the dominant technology. Minicomputers introduced the first of a set of alternative technologies that eventually evolved into client/server computing. Today, the benefits of centralized computing are rematerializing as large organizations consolidate servers and turn to running heterogeneous applications and OSs on mainframes. The return to mainframes, while enabled by technical advances such as Linux on the mainframe, is driven by fundamental business objectives such as cost control.

ECM is one of the fundamental technologies for ensuring organizations meet their objectives. Disciplined change management enables essential management objectives such as:

- Proactive control of change in the organization
- Insight into the impact of change
- A basis for prioritizing changes

Throughout this book, I will discuss specific elements of ECM, such as the software development lifecycle, hardware configuration management, and document control. All of these factors contribute to the overall objective of harnessing change to advance the objectives of the enterprise.

Chapter 2: Examining the Nature of Enterprise Change Management

The need for change within organizations comes from many sources. Innovative technologies, market pressures, and regulatory changes are just a few. These drivers affect organizations in a variety of ways; many set in motion complex sequences of change that propagate effects far beyond the original point. As we explored in Chapter 1, a simple upgrade in a word processor program can change the way staff collaborate and share documents as well as introduce security vulnerabilities that eventually lead to a widespread disruption. This chapter will examine the nature of ECM by breaking down complex processes into constituent parts and analyzing their interactions.

I'll begin by describing a generic model of enterprise change and showing how that model provides a guide to understanding ECM tools. The sections that follow discuss the process of managing enterprise change and examine its lifecycle. Finally, the chapter concludes with a discussion of the need for domain-specific additions to the generic model for software development, systems configuration, and document management. Each of these three domains will be examined in more detail in Chapters 3 through 5.

Modeling Enterprise Change

Although we can talk about changes at the enterprise level in broad terms, it is more useful to start with a more precise discussion. To understand the nature of enterprise change, we need to identify specific *assets* that change and understand how the changes in one asset modify another. For example, specific assets include software packages, Web services, networking hardware, security policies, manufacturing processes, and marketing initiatives. Each of these assets has both static attributes and dynamic lifecycles.

Attributes are general characteristics of a resource that you use to describe the resource. An instance of a CRM system, for example, runs on a particular platform, has a specific version, and uses a relational database for storage. Similarly, marketing initiatives are associated with product lines, have budgets, and have executive sponsors. It is likely that during the life of the CRM application, the application will be upgraded to a new version or migrated to a different platform. While executing a marketing initiative, the budget might change in response to an unexpected downturn in the market. These changes in the core attributes of an application or marketing initiative characterize the *lifecycle* of the resource.

Lifecycles, like attributes, vary among resources. An application's lifecycle entails development, testing, release, maintenance, and retirement. Documents are created, reviewed, approved, published, revised, translated, and archived. Hardware systems are evaluated, installed, integrated, monitored, and replaced. Strategic initiatives are formulated, revised, approved, implemented, and evaluated. Fortunately for those who manage change, the lifecycles are generally predictable, moving from one state to a small set of other states as Figure 2.1 illustrates.

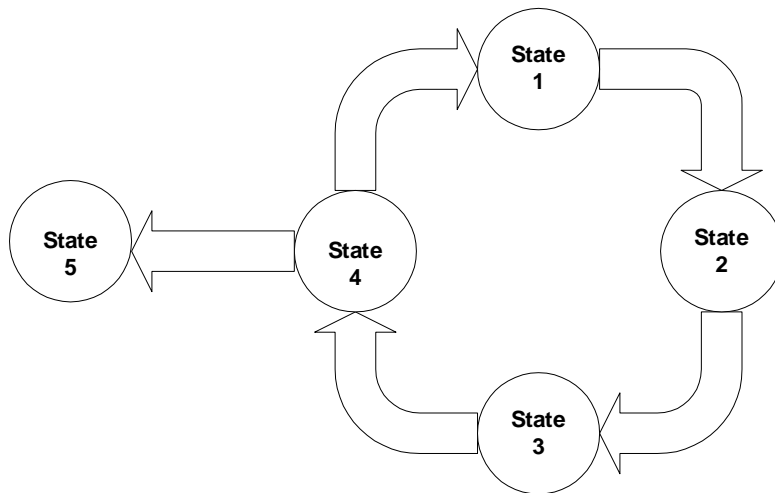


Figure 2.1: The lifecycle of resources follows a predictable pattern although multiple outcomes are possible from any given state.

Individual assets, like individual organisms in an ecosystem, have lifecycles that interact with others. To understand the dynamics of a complex system, whether it is a biological ecosystem or an enterprise IT infrastructure, you have to understand how lifecycles influence each other. For example, an increase in a fox population leads to a decrease in a rabbit population. However, as the rabbit population declines, the food source for the foxes declines leading to a reduction in the fox population. The declining fox population decreases the threat to the rabbit population, which, in turn, will grow over time. The changes in one population directly affect the other. Similar effects are seen in organizations. As enterprise assets change with events in their lifecycles, those events trigger actions in other resources' lifecycles (see Figure 2.2).

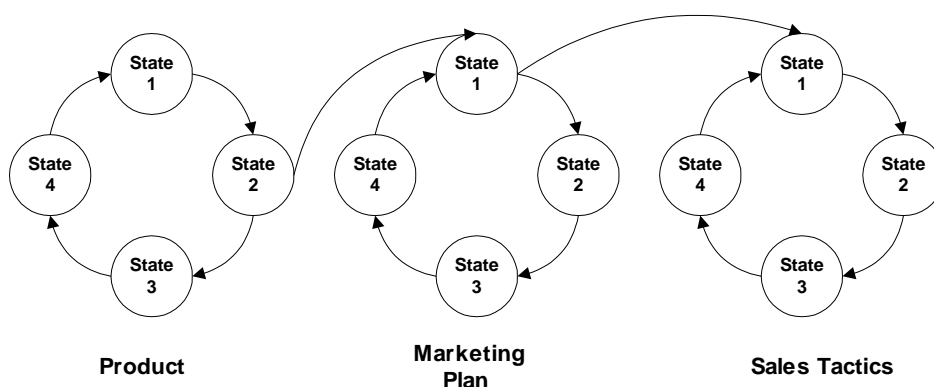


Figure 2.2: Events in the lifecycle of one event can initiate changes in the lifecycle of other resources.

For example, suppose an LOB manager decides to reposition a product line to take advantage of a new trend in the market. The marketing department updates the marketing plan, which is then passed on to regional sales directors who revise their sales tactics. They, in turn, inform their sales teams who change how they present the product to prospective customers.

Nature has numerous mechanisms propagating change throughout an ecosystem—populations migrate, species adapt, and animals change behavior according to their situations. Enterprise ecosystems, especially IT ecosystems, do not adapt as gracefully as biological ecosystems. Changes in one resource can disrupt services, degrade performance of other systems, and force unwanted changes on operations. When changes are not communicated, customers and partners receive inconsistent information and service. Rather than depend upon a nature-like cycle of change and adaptation, dynamic organizations need more centralized, proactive controls over enterprise change. The first step to developing those controls is modeling their requirements.

Constructing a Model of Enterprise Change

Models make complex systems understandable. They also allow us to see patterns that are not necessarily obvious when we see systems in detail. The characteristics shared by the processes of introducing a new version of an application, a policy document, and a network device are often not obvious—especially when we think of common domains such as software, documents, and networks. For example, when we think of software applications, many of us immediately think of programming languages, software design patterns, version control systems, release management policies, and a host of other software engineering techniques and tools. Similarly, when a new policy document is introduced, we think of document revisions, metadata, approval workflows, publication processes, and other document management tasks. Network device configurations bring to mind protocols, security issues, network outages, and hardware compatibility. Identifying the unique and obvious characteristics associated with each domain is quite easy compared with identifying the more nuanced but equally important shared characteristics of each process. Without an understanding of these inter-relationships, organizations tend to rely on silo-oriented change-management procedures. For each domain, IT staff has particular methods for dealing with change rather than using a single approach for change management across the enterprise.

Complex organizations need to extend these change-management techniques to better work across functional domains. To begin, IT staff needs to understand the similarities between functional domains (such as software development, system configuration management, and document management) and describe them with models.



Recognizing *shared* characteristics of different change-management domains is the foundation of modeling ECM.

Managing any operation or process requires that we understand its essential elements. Models help us do so by distilling processes to their most basic components and operations. Many of us are familiar with some change-management methods for particular domains (such as software configuration management) and not with others (such as document management). These domains share common characteristics, and a generic model helps to highlight those properties. In addition, a generic model provides a reference for evaluating and comparing ECM tools. ECM models are made up of five components:

- Assets
- Dependencies
- Policies
- Roles
- Workflows

Assets

Assets are objects that change. Software, policies, network devices, strategic plans, and marketing initiatives are all examples of assets. All assets in ECM models have several characteristics:

- Versions
- Issues
- Attributes

Assets change over time, and versions track logical groups of changes. In the case of software, documents, and other digital assets, organizations frequently keep older versions as well as the latest release; however, only a single version of a hardware asset usually exists. For example, an asset called Boston Router 2 might not be the same physical device today that it was 4 years ago, and the earlier version may have been redeployed, sold, or disposed of.



How versions are realized (such as with a new file or a new piece of hardware) is unimportant for the model. ECM models represent the shared characteristics of assets.

Issues, as you might have guessed, are topics about an asset that must be addressed. Defects, system incompatibilities, missing features, proposed content changes, and legal questions about policies are examples of issues. Again, the differences in the types of issues are unimportant. Software has bugs; documents do not. Documents have copyrights; servers do not. Nonetheless, the general property of tracking issues applies to resources that change over time.

Attributes are characteristics that help describe an asset. (Versions and issues are attributes in the strictest sense. For our purposes, however, consider attributes to be the set of characteristics that change with asset type. All assets should be modeled with versions and issue attributes.) Documents kept in a content-management system will have a creation date, an archive date, key terms, categories, summaries, language, description of rights, and access controls. Software modules have major and minor release numbers, and a description of change history that includes the name of the programmer, the type of change, the date of the quality control check, and the release date. To model change, you don't need to know the exact list of attributes for each asset type you want to manage. This freedom is one of the benefits of a model of ECM—it allows you to focus on the unchanging, core properties of enterprise change, and leave the varying details until later.

Dependencies

Dependencies identify assets that are required by other assets. When modeling dependencies, you need to consider the granularity of assets and the types of dependencies.

Assets can be described at a number of levels. At a high level, you can model an ERP system. A more detailed model breaks the ERP system into modules such as inventory, scheduling, and database. Each of these components can be further decomposed. For example, the database module includes the relational database engine, backup software, performance monitoring tools, and network client software.

So what is the appropriate level of granularity for describing assets? In general, assets should be modeled at the level that dependencies apply to all sub-modules of an asset. Consider Asset A, which consists of Modules 1, 2, and 3. Asset B depends upon Modules 2 and 3 of Asset A. Figure 2.3 shows the appropriate level of granularity.

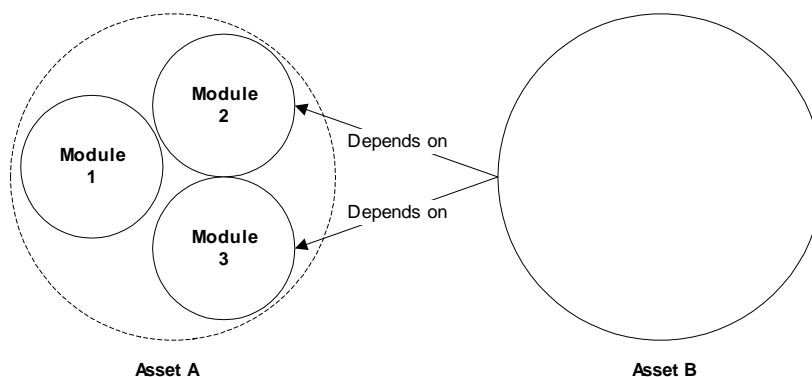


Figure 2.3: Asset granularities should be chosen to precisely depict dependencies between assets.

Models that do not precisely describe dependencies can misrepresent relationships by depicting them as more restrictive than they are. For example, Figure 2.4 shows an imprecise, or course-grained, dependency model that inaccurately depicts Asset B as dependent upon all three modules, when, in reality, it doesn't depend on Module 1.

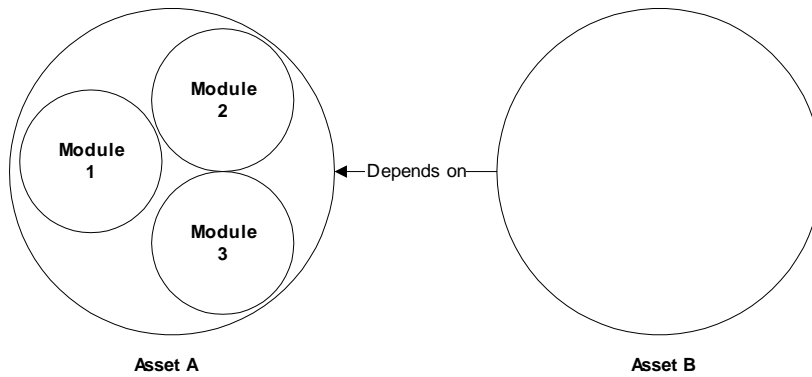


Figure 2.4: Course-grained models can erroneously depict dependencies.

Similarly, models that are too fine-grained will depict dependencies in a more complex manner than necessary. If Asset B depends upon Modules 1, 2, and 3, then Figure 2.4 depicts the appropriate level of granularity; Figure 2.5 depicts a version that is too fine-grained, which creates more work than necessary to accurately model dependencies (most models will be much more complex than this simple example).

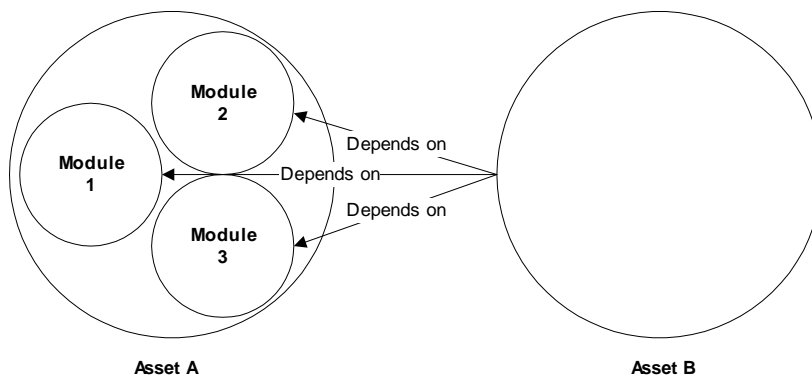



Figure 2.5: Because Asset B depends on all modules of Asset A, this model is too fine-grained.

Policies

Policies describe the rules that govern the use of assets and the mechanisms of change. Some policies are asset-specific, such as procedures for introducing a new piece of network equipment, creating a new branch in a software configuration management system, or approving changes to an HR policy. Other policies are independent of assets. For example, rules about notifying others about changes and workflow approval policies are not restricted to particular types of assets.

 Policies themselves have change-management issues. The lifecycle of a policy should be governed by change-management procedures.

Policies have similar lifecycles to other enterprise assets. For example, a change in federal regulations may cause an organization to change its policies about sharing customer data. This external change forces the company to revise its policy, which, in turn, creates changes to policy content. The change in content initiates a workflow of reviews, approvals, and publishing that ultimately requires communicating information about the new policy (see Figure 2.6).

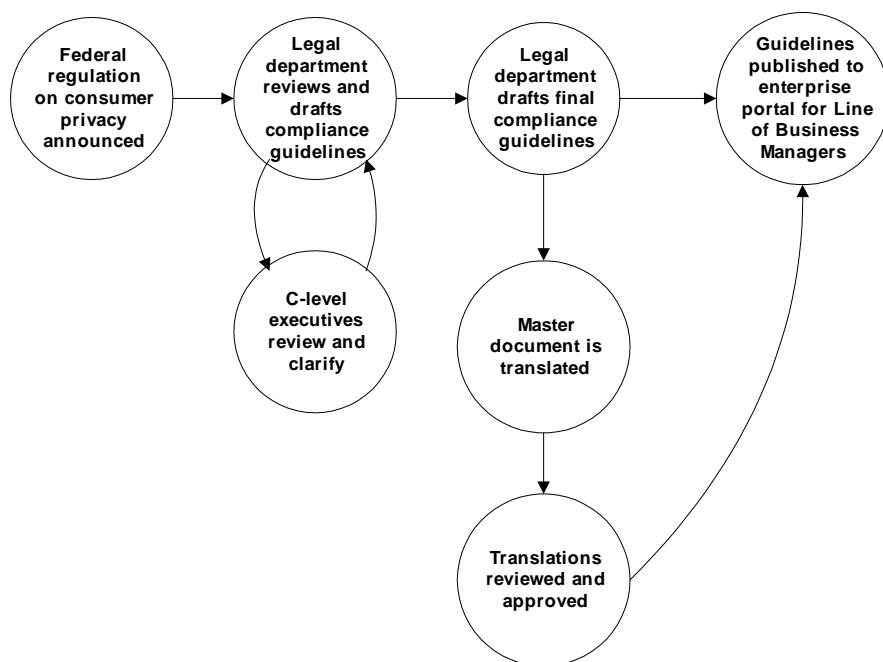


Figure 2.6: Changes to policies are also managed through controlled workflows.

Roles

Roles are assigned to individuals or groups who make decisions and execute workflows. These people follow the rules described in policies to create and maintain assets and to control change across the enterprise by:

- Approving change
- Deciding on the specifics of a change
- Reviewing a proposed change
- Preparing for change
- Implementing change

Roles are frequently used in security systems to control access to resources and serve a similar purpose in change-management systems.

Workflows

Workflows are processes that change the state of assets and related objects, such as change request forms. Workflows manage the change process, coordinate people and assets, and enforce policies. The purpose of workflows is to enable a consistent, rule-based process for executing change. At its most basic level, workflows entail:

- An initial start state
- Rule-based transitions to intermediate states
- One or more end states

The initial state is the activity that starts the workflow process. This activity can range from creating a change request to installing a desktop application to compiling a CRM upgrade proposal for an enterprise change review board. Workflow processes move from initial states to intermediate states according to transition rules.

Transition rules are conditional requirements that, when met, change the state of a workflow. For example, *If the change request to install a desktop application is approved by the support desk supervisor, the request is added to the job queue* is an example of a simple transition rule. Once in the new state, a new set of transition rules applies. For example, *If the job is to install a new desktop application, the priority is 4 and Assign jobs to support desk personnel according to increasing priority* are two rules for controlling transitions to other states.

A workflow is finished when the end state is reached, such as when an application is installed, documentation is updated, and acceptance tests are passed. For example, *Notify Requestor of Status* is the end state of the workflow illustrated in Figure 2.7.

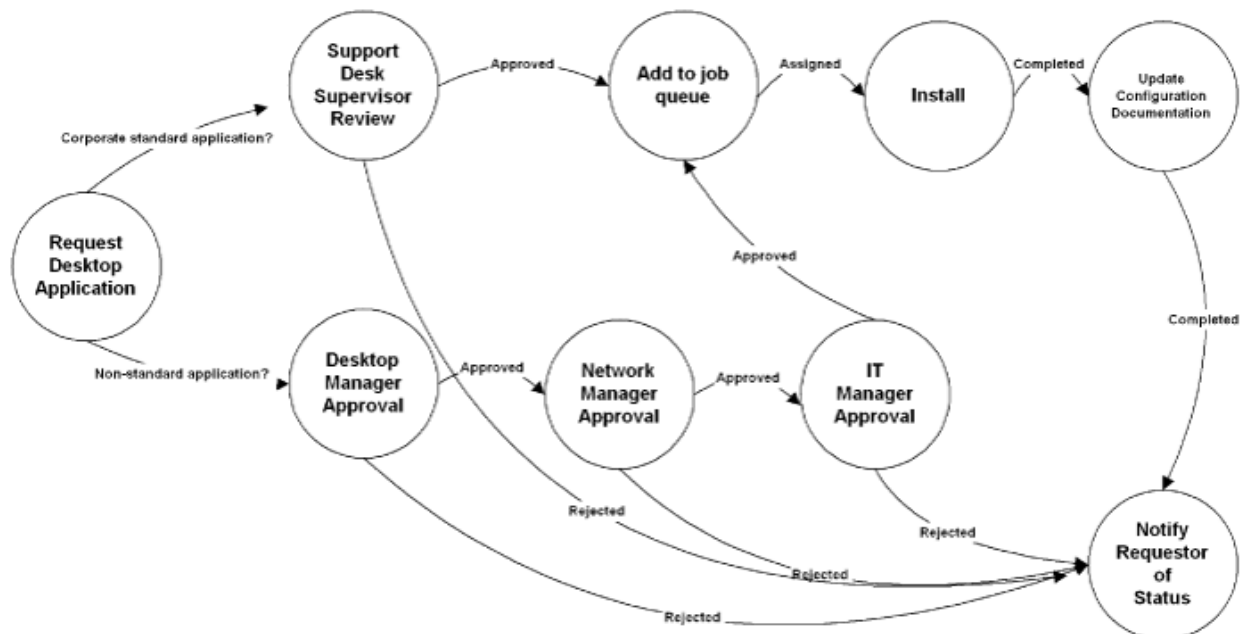


Figure 2.7: Workflows consist of states and transitions between states.

The generic model of enterprise change is made up of assets, dependencies, roles, policies, and workflows. By themselves, they do not completely describe the ideal ECM application; however, they give us a tool for evaluating and comparing such tools.

Using the Reference Model to Define ECM Applications

We need to remember that the need for change management encompasses virtually all parts of an organization. Changes in one department or application can quickly ripple through to other areas and systems. To compound the complexity, change crosses functional divisions, so a change in one software module can impact other software modules, network configurations, and documentation. Developing a model of enterprise change is the first step to creating an ECM management system. The second step is to figure out how to implement that model as an ECM support system.

Functional Characteristics of ECM Systems

Managing change in an enterprise requires three elements:


- Tight communication
- Asset metadata repositories
- Workflow management

We've already discussed workflow management, so in this section, I'll focus on the other two elements.

Communicating About Change

It is clear that changes emerge from multiple sources. A user may request a feature change to a program. A merger introduces new hardware to a WAN. New government regulations mandate additional procedures for product development and production. When a change is communicated in advance of its implementation—for example, when a user requests change to a program—the change is readily managed. But when the impact of a change is discovered after implementation, such as when the upgrade of an enterprise application creates unanticipated demand on network bandwidth, the consequences are more difficult to manage.

To control change, managers and administrators need to understand the impact of external changes to their internal operations. The first step in this process is being aware of the specific details about forthcoming changes. This, in turn, requires that those who are introducing changes understand the scope of their modifications and the potential impact. For example, a Web server administrator may not realize that upgrading to a new release of the Apache Web server will cause a business intelligence reporting tool to crash. The Web server administrator needs a tool that identifies such a dependency. This type of situation is where asset metadata repository (AMR) comes in.

 Dependencies between resources are too complex to manage manually. ECM requires tools for tracking dependencies between resources.

Tracking Changes with AMRs

AMRs track information about individual assets and dependencies between them. As noted earlier, assets are described by attributes. Assets often require other assets to function properly, as Figure 2.8 shows. ERPs depend upon databases. Databases require OSs and servers. OSs depend on networking hardware to communicate with other servers, such as domain controllers.

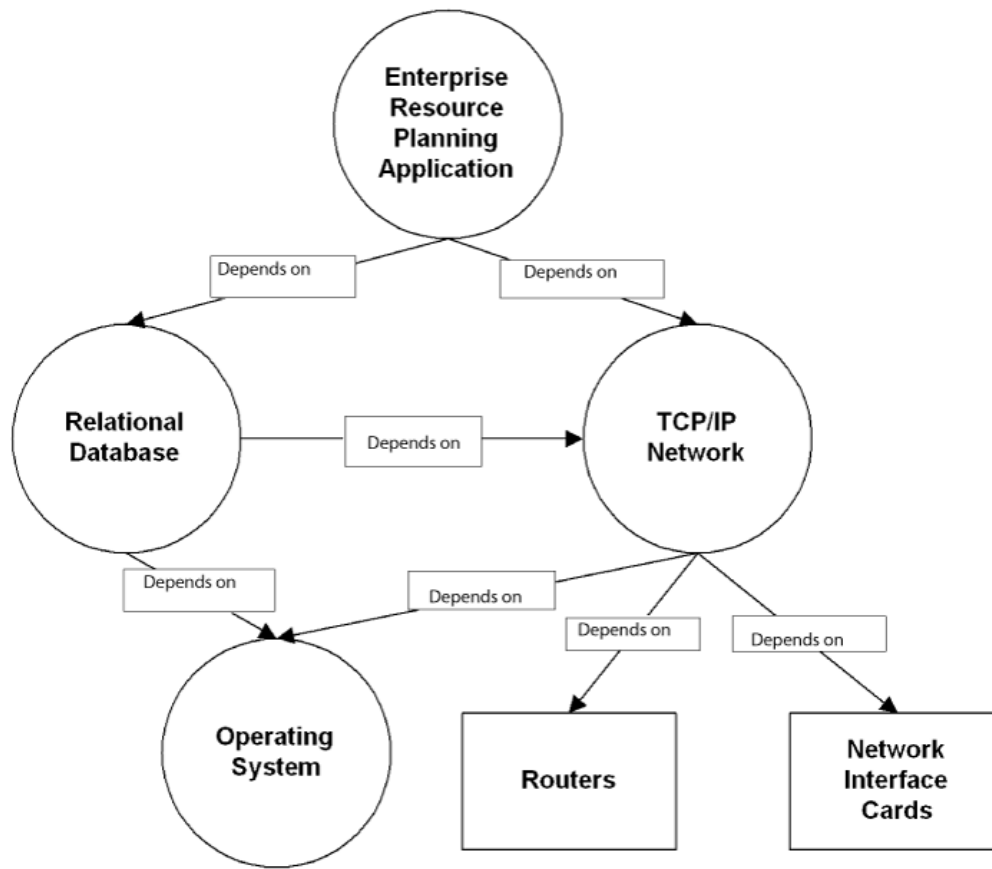



Figure 2.8: Resources can depend upon layers of other resources.

Dependencies are logical relationships between one asset and another, or more specifically, between one asset and a property of another asset. For example, the latest version of an ad hoc reporting tool, Reporter X, requires portal software, PortalSoft Y, version 2.3 or later. Dependencies imply constraints not only on the type of an asset, but can restrict particular attribute values as well.

Dependencies vary by type. One way to categorize dependencies uses the following three types:

- Requires—An asset needs another asset with particular properties to execute correctly
- Consumes—An asset uses the output of another asset
- Supplies—An asset produces data used by another asset

By keeping track of assets and their dependencies on other resources, AMRs enable administrators and managers to perform impact analysis of proposed changes. For example, consider an administrator who wants to upgrade a server to use the latest version of the Linux kernel. The new kernel provides performance enhancements that will benefit all applications, but the kernel also contains changes to code libraries that could adversely affect some applications. The administrator wants to know which applications on the server are affected. Using an AMR, the administrator can list the applications running on that server and the dependencies of each application.

 AMRs are essential to answering the question, what is affected by a particular change?

Let's assume that a machine is a dedicated ERP server that is compatible with the latest Linux kernel, so there should be no problem with the upgrade with the first level of dependencies. We need to keep evaluating the effects of change on deeper levels, though. The ERP server depends on an Oracle database that has not been certified with the latest Linux kernel. However, the database runs on a different server, so it is not affected by the change. Or is it? Although the ERP and database run on different servers, communication software runs on both to enable the distributed system. The database communication software on the ERP server has dependencies on the OS. Although the overall database system is not affected by the OS change, a small, client module is affected. To manage fine-grained dependencies such as this, the AMR should model details of individual modules within larger applications (see Figure 2.9).

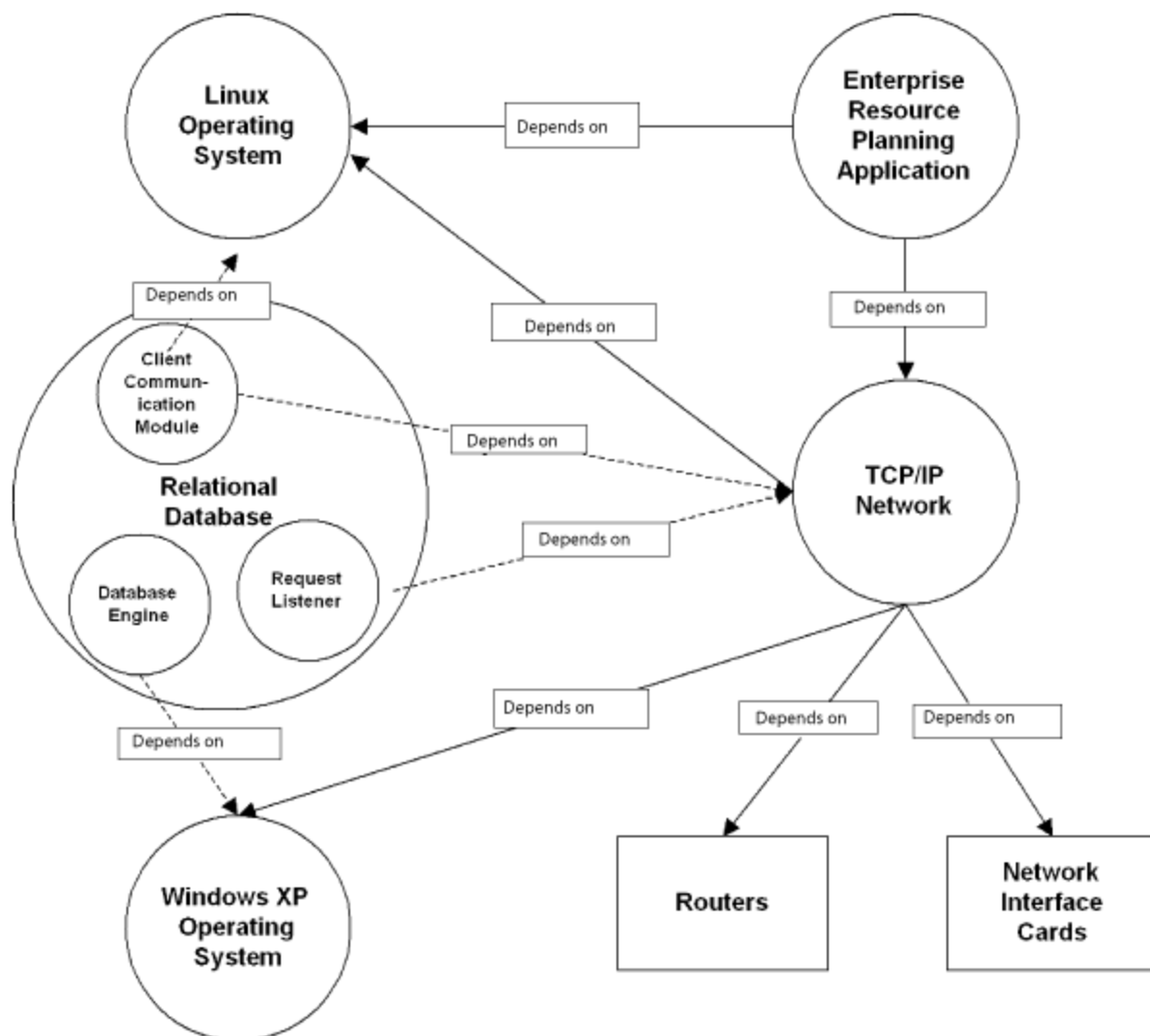


Figure 2.9: AMRs should model fine-grained dependencies—especially in distributed systems.

Tight communication is essential to understanding upcoming changes that affect other parts of the organization. AMRs manage information about enterprise resources and the dependencies between them. Workflow management systems coordinate tasks between people and automated processes. Together, these three elements act as the foundation for managing change, as Figure 2.10 shows.

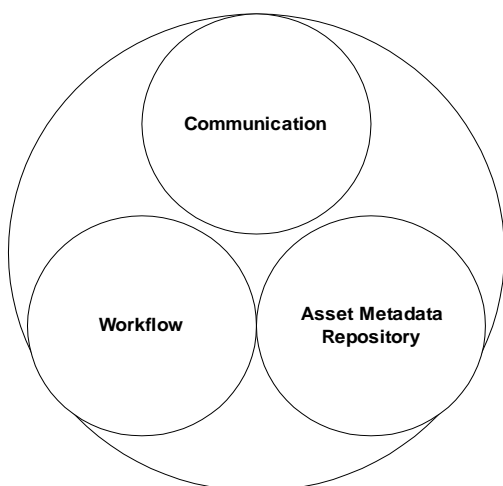


Figure 2.10: The ideal ECM environment consists of AMRs and workflow processes that support tight communication.

With an understanding of the mechanisms required to control change, let's turn our attention to the design characteristics of the ideal ECM system.

Implementation Characteristics of ECM Systems

ECM is a dynamic system that inextricably links process and data. To support the breadth and complexity of change found in large organizations, ECM systems must deeply integrate processes and data. This requires:


- Bi-directional synchronization at the process and data levels
- A shared common repository
- A shared common user experience

Synchronizing Process and Data

A common operation in the ECM is defining workflows. These processes require that you specify assets and rules for manipulating those assets. Once defined, a change to either the workflow process or the assets can affect the other. For example, consider the following business rule: *If a program is on the list of corporate approved applications, the program can be installed on client machines with the approval of the support desk supervisor.*

Let's assume that users submit change requests through an online system that automatically routes requests through a workflow process. When a user requests an installation of a word processing program, the workflow system will need to check the AMR to determine whether the program is on the list of corporate-approved applications.

Similarly, if the workflow process changes the property of an asset, the AMR should be updated. For example, if a project manager approves the release of a new revision to a software module, the repository is updated to include a record of the new version and its properties. The workflow mechanism and the AMR are closely linked in their basic operations (see Figure 2.11).

 ECM depends upon deep integration of workflow and metadata about assets.

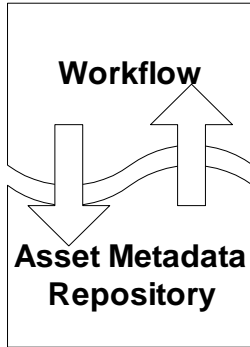


Figure 2.11: Workflow processes depend upon the AMR, and the AMR is updated by workflow operations.

Using a Shared Metadata Repository

The AMR is a common asset repository. Information about software, hardware, network configuration, and content is kept in the AMR. This shared repository design enables deeper integration of ECM than individual repositories. For example, consider the design that Figure 2.12 shows. Silo-based repositories—such as this design—lead to well-known problems, such as inconsistencies among the data and the need for ad hoc data exchange mechanisms.

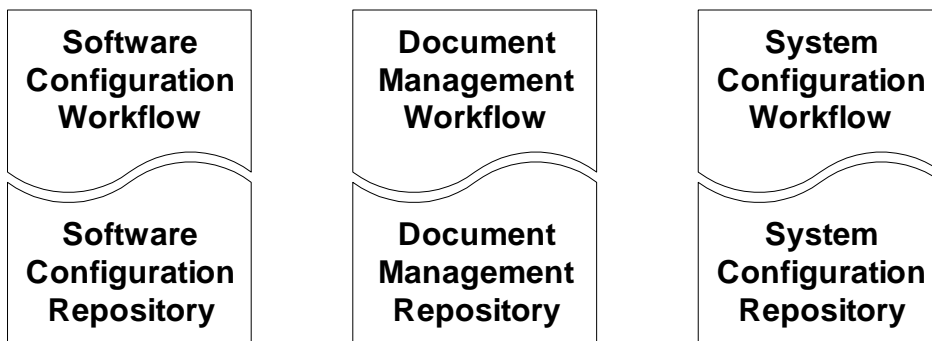


Figure 2.12: Silo-based configuration management is not an effective option for ECM.

Ideally, releasing a new software module will initiate or correspond with a change in documentation about the module. When the software configuration management workflow and the document management workflow are independent, this change is difficult to automate. The result is a potential inconsistency between the repositories and a break in the logical chain of ripple effects caused by the initial change.

Sharing the User Interface

Another benefit of the deep integration is that it allows for a shared user experience. Users learn a single interface and work in a single environment. Of course, some operations are specific to particular types of change. Documents have keywords and authors; network hardware does not. Nonetheless, by using the model of ECM described in this chapter, you can model different assets and operations within a single application and provide the users with a consistent and unified experience.

Users benefit from both cross-functional change management and the ability to work within a single environment. The single model design also reduces the number of applications that must be integrated with the existing IT infrastructure. The more the ECM system, especially the workflow elements, can be directly integrated with other applications, the more you can automate the change-management process. With an understanding of the static elements of ECM systems, let us turn our attention to the more dynamic characteristics of those applications.

The Enterprise Change Lifecycle

The three elements—tight communication, AMRs, and workflow management—describe the anatomy of ECM. The lifecycle of ECM consists of four stages:

- Initiating change
- Reviewing change
- Implementing change
- Assessing change

The first stage of the lifecycle is initiating change. This stage could entail creating a new asset, such as a policy document; changing the configuration of an asset, such as a router; or creating a new version of an asset, such as a revised program. Proposed changes are then reviewed. This process can range from rapid, single-person review (such as approving the installation of a desktop application) to more formal and detailed plan reviews by a change-control committee. Once approved, the change is implemented following procedures specific to the type of change and type of asset. Finally, the effects of the change are assessed and are either left in place or, if unanticipated consequences emerge, rolled back. Figure 2.13 illustrates this concept.




Figure 2.13: *The four-stage enterprise change lifecycle.*

With an understanding of the general elements and lifecycle of the ECM model, let's shift our focus to specialized types of change management.

Specializing Change-Management Operations

To this point in the chapter, I have described a generic model for enterprise change and discussed how several basic elements and processes can support a wide array of changes. This design works well as a reference model for understanding the nature of ECM and its basic elements (assets, roles, policies, and so on). To be truly useful in day-to-day operations, however, we need to support the particular needs of specific processes and assets.

In this section, I will further expand the ECM model to include characteristics of software, system configuration, and document change management. I chose these three types because they are commonly required and highly interdependent. We can imagine higher-level asset types, such as strategic initiatives and marketing campaigns, that build upon these and require change-control mechanisms as well; however, for clarity, I'll keep the focus limited.

 The ECM model described here is generic enough to apply to other organizational processes that entail assets, processes, and dependencies.

Identifying Software Configuration Management Requirements

The requirements for software configuration management are driven by the unique lifecycle of software. Large software systems are modularized to control complexity. The number of modules and the levels of dependencies can grow quickly. Software engineering practices promote code reuse, so a module designed for one application may be used in several others. Modularization and reuse force, at least implicitly, agreements between module developers and module users.

The simplest module is a function that returns some data based upon other data provided as input. Users often make assumptions especially when dealing with simple functions. For example:

- The function does not have side effects (such as changing a global variable or updating a database record)
- The function accepts a reasonable range of inputs (such as 0 to 110 for a customer's age)
- The code will degrade gracefully if there is an error (such as returning an error status instead of crashing the process)

These are reasonably safe assumptions because they fit with common programming practices—especially when dealing with simple functions. As functionality becomes more complex, behavior of the module must be made explicit through detailed documentation.

Explicit descriptions of behavior allow software developers to effectively combine modules to build complex systems in a layered manner. As Figure 2.14 illustrates, a programmer responsible for developing the user interface (UI) might need to work with procedures in the data access layer, but the developer should not have to work directly with database or network components. However, the interface code still depends upon those lower-level modules. Changes to those modules can directly impact the interface.

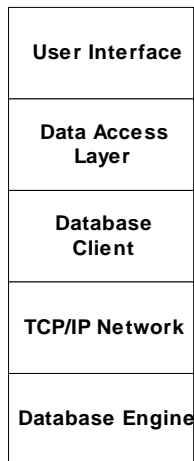



Figure 2.14: Modularized and reused code creates indirect dependencies between modules in complex systems.

Software engineers have developed configuration management practices to manage change in complex software systems. Typical development teams use version control, system builds, and system releases to control change.


 Software engineers that change code need to understand not only what the program does but also where and how it is used. The AMR tracks the required information.

Version control, as the name implies, is a mechanism for tracking changes to program code and documentation. Programmers check out code from a repository to work on the code. While the code is checked out, others are prevented from changing the code in the repository. The programmer checks in the code when he or she has finished making changes and creates a new version of the code. Although this technique keeps developers from overwriting others' changes, it is not enough to ensure that a change does not adversely impact a dependent module.

To ensure that programs continue to work as expected, software developers frequently combine the latest versions of all modules and test the results in the build process. This step consists of compiling source code, linking libraries, executing test suites, and comparing the test results with those of previous builds.

Once the system is built and tested, it is released for use. Release management practices govern how executable code is distributed to users, how dependent modules—such as OS components, link libraries, and Web services—are put in place, and how users are trained to use the new application.

Like other types of change, software configuration management is highly dependent upon rule-based workflows. Programmers check in and check out code, project managers approve code for inclusion in a build, quality control analysts evaluate test results, and release engineers roll out the application. System configuration management, like software configuration, has its own particular issues.


 We'll explore software configuration management in more detail in Chapter 3.

Identifying System Configuration Issues

One of the trickier aspects of system configuration management is continuing to provide services while making changes to the IT infrastructure. If a network engineer needs to change a router, another router must be in place to keep network traffic flowing. Large organizations often have dual ISPs. Critical servers are configured to failover to backup servers that run in stand-by mode. Redundancy is common in hardware systems, and change-management procedures need to account for them. Some characteristics particular to system configuration include:

- Identifying backup systems
- Describing how failover occurs (for example, automatically in the case of routers, or manually in the case of offline servers brought online only after a failure)
- Describing clusters and load-balancing configurations

Security issues might be tracked independently of particular assets. Of course, one can track security issues of a particular device or software (such as a bug in a firewall or router). In other cases, security issues arise as a function of two or more assets operating together.

 In Chapter 4, I'll discuss system configuration management in greater depth.

Identifying Content and Document Management Issues

The most commonly encountered change-management issues in content and document management are:


- Archiving
- Indexing
- Publishing
- Translating
- Access controls
- Reuse and structuring

As the number of documents grows in a document management system, the task of administering the repository becomes more difficult. To ease some of the burden, many administrators use archive dates to indicate when documents should be removed from the repository. Archive dates can be specified manually along with other metadata or automatically generated using simple rules (such as completion date plus 2 years) or with more complex business rules based upon the type of document, the author, the category of content, and other metadata attributes. For example, 21 CFR Part 11 regulations governing pharmaceutical manufacturing processes dictate strict requirements on records retention.

When the content or metadata attributes of a document change, the document repository index must be updated. This task includes updating automatically generated categorizations. If there are significant changes to a repository, such as adding documents about new topics, the taxonomy or other classification scheme will require updating as well.

Changes in a content management system may also trigger the publication of documents in multiple formats. Interested users might automatically receive copies of or notifications about the new documents. In multi-language environments, translations must be revised to reflect changes in the source content. For example, consider a global document management system for product documentation. Master documents are written and revised in the company's primary language, then translated into other languages. Each translation leads to a new version of the document that must, in turn, undergo review and approval procedures defined in a workflow.


As with any IT system, the particular requirements for access controls will change over time in a content management system. Updates are driven by both changes in the user base and changes in requirements to protect different document classes. For example, the state of California has enacted a law requiring companies to notify California customers when a security breach leads to a disclosure of personal information. This requirement may prompt companies to change data management procedures for California customers, move California customers' data to separate repositories, or even lead some to stop conducting business in that state.

 Supporting multiple languages and maintaining effective security are two of the most challenging aspects of document management.

Reuse and restructuring of documents is much easier now than in the past. Extensible Markup Language (XML) is radically changing content management. Prior to the widespread adoption of markup languages, such as XML and HTML, documents were treated as single, logical units. A short memo and a 300-page budget were both stored as a single document. (Large documents, such as books, could be broken down into a series of files stored in directories, but this approach still retained many disadvantages.) These documents are generally referred to as unstructured text because the documents are not structured like databases or spreadsheets with easily identifiable pieces of data. From a content management perspective, unstructured documents have two drawbacks: they make it difficult to reuse content and they only provide document-level metadata.

Unstructured documents make it difficult to reuse content. For example, a product brochure might contain a brief description of a product that could be used in an online catalog. Traditionally, that description was cut from the brochure document and pasted into a description field for the product in a content management system. For companies with a global market, the problem is compounded with the need to reuse multiple versions of translated content.

Another drawback is that unstructured documents only provide document-level metadata. For example, the metadata about a product brochure document might describe the author, creation date, version number, keywords, and document description. Unstructured documents do not specify details such as where the title begins and ends, where the short product description begins and ends, where pricing information begins and ends, and so on. Without complex programs to analyze the text, it is impossible to reliably extract pieces of information from unstructured documents.

 For more information about the role of XML and related technologies to business, see the Organization for the Advancement of Structured Information Standards Web site (<http://www.oasis-open.org/home/index.php>) and for more technical details, see XML.org (<http://www.xml.org>).

For these and other reasons, content managers are turning to XML and related technologies to better manage content. The most important are XML, XPath, and XQuery.

XML is a markup language that lets you structure parts of a document. The documents still contain free form text—they are not as structured as databases or spreadsheets—so they are called *semi-structured documents*. For example, a simple product description in an unstructured document follows:

```

WP 1234 Wireless Phone

"The WP1234 2.4GHz cordless phone offers caller-id, call back,
100 number memory, speaker phone ..."

"Recognized for superior sound quality and reliability the WP
1234 ..."

```

A corresponding XML document adds structural information and metadata such as the example in Listing 2.1.

```


<product-description>
  <product-name> WP1234 </product-name>
  <features>
    <feature>2.GHz</feature>
    <feature>caller-id</feature>
    <feature>100 number memory</feature>
    <feature>speaker phone</feature>
  </features>
  <short-description>The WP 1234 2.4 GHz cordless ... </short-
description>
  <long-description>Recognized for superior sound quality and
reliability,
the WP 1234 ...
  </long-description>
</product-description>

```

Listing 2.1: An example XML document.

The semi-structured XML version includes the same information as the unstructured file as well as indicators, or *tags*, that identify specific information such as the product name and a short description. These additional tags make the files more difficult to read for humans but greatly enhance our ability to automatically process these files. XML technologies, such as XPath and XQuery, provide the means to identify a particular piece of information in a document (for example, the short description), then query for it much like we query databases.

As XML, XPath, XQuery, and related standards are adopted, organizations will have more opportunities to reuse document components for multiple purposes. For example, a single product description maintained in an XML database may be used in brochures, catalogues, and email solicitations. (Identifying and extracting appropriate content requires managing the XML schemas, which, in turn, introduces change-management issues. There is no end to the need for change management).

 We'll explore content and document change management in detail in Chapter 5.

Clearly, there are similarities and differences across domains in modern enterprises. Effective ECM tools will support both.

Supporting Enterprise Operations with Change-Management Services

Change management is evolving. Understanding and managing change is becoming more complex as businesses integrate operations and cross-traditional functional boundaries. One of the most important aspects of ECM is the interdependence of changes. A change in one type of asset often triggers a change in another type of asset. A change in one department or line of business frequently initiates a change in the operations of another department or functional area. Analyzing and planning for the scope of change that ripples through an organization when silo-based change-management systems are used is challenging at best. These systems do not provide the broad, cross-functional impact analysis that is needed to effectively manage enterprise change.


Impact analysis allows managers to model the effects of a change. Consider the following examples:

- A change in a client application requires changes to the hardware configuration of client workstations.
- An upgrade to an application server will not function properly without reconfiguring a firewall and the virtual private network (VPN).
- Changing a marketing plan will change the product presentation for the sale staff.


To understand the detailed effects of these changes, managers must have access to configuration and dependency information. That data, along with impact analysis algorithms that trace paths of dependencies through networks of assets, are the key to impact analysis reporting. Impact analysis allows administrators to conduct what-if planning but can also help determine:

- Who to notify about forthcoming changes (for example, other systems administrators and application managers)
- When to implement a change to minimize down time
- When to reconfigure services to prevent a service outage
- What changes to hardware are required
- If there are any incompatibilities with layers of software
- Any potential increase in calls to the Help desk

The benefits of ECM extend beyond impact analysis. These techniques allow organizations to maintain compliance with auditable standards, such as HIPAA in the healthcare industry, 21 CFR Part 11 in pharmaceuticals, the Gramm Leach Bliley Act in financial services, and the Sarbanes Oxley Act in publicly traded companies.

 See Chapter 1 for more details about how external requirements such as audit standards create the need for ECM.

As studies of the use of software capability maturity models have demonstrated, controlled methodologies improve the quality of application development. Although it is too early to measure the impact of ECM techniques, it's reasonable to expect similar levels of improvement in non-software development areas.

 For more information about software engineering and capability maturity models, see the Capability Maturity Model Integration Web site (<http://www.sei.cmu.edu/cmmi/>).

ECM procedures provide C-level executives with enforceable policies and procedures. They provide impact analysis tools for planning and managing change. In short, they enable proactive planning and serve to minimize the need for post-change damage control.

Summary

Organizations are constantly changing to adapt to new situations. The process of continual change is a fundamental process for large enterprises and cannot be avoided. Even if an ideal internal configuration for an organization could be found, external factors, originating from customers, competitors, and governments, will change the environment in which the organization operates.

Change-management practices have long improved the quality of software development, system configuration, and document management. To move beyond domain-specific techniques, we need a generic model of enterprise change. A basic model, like the one described in this chapter, helps to elucidate similarities between change-management domains, which, in turn, provides a foundation for understanding how to manage changes that cross domains.

As organizations move from silo-based change-management techniques to ECM, they will need systems that describe assets and their dependencies, policies and roles for controlling assets, and workflows to use assets. Organizations will also need to understand the lifecycle of change. Enterprises are like ecosystems—changes in one part of the environment have ripple effects on distant parts. Unlike natural ecosystems, today's organizations cannot sustain uncontrolled change.

Organizations control change with tools that support tight communication and workflows using AMRs. The tools we need are evolving along with the need for ECM. If you are assessing ECM tools, use the generic model described here as a reference point. The generic model might not describe all features required to control change, but it does include the minimum functionality needed to support ECM.

In the next three chapters, we will use the generic model to discuss how to effectively manage change in software development, system configuration, and document management, respectively. At the same time, we will examine how changes in these domains cause change in other domains, and how you can control that cross-domain change.

Chapter 3: Managing Change in the Software Development Lifecycle

Software development is a task common to most enterprises—even organizations that are not in the software development business, such as financial institutions, manufacturers, and government agencies, design and develop software applications for internal use. The process of developing software follows a standard lifecycle, and, as we'll explore in this chapter, managing that lifecycle is an integral part of ECM.

Levels of Change Management in Software Development

Software developers depend upon change-management practices at several levels: individual, team, enterprise, and across organizations. At the individual level, programmers and designers manage their code and documentation in isolation, then share their programs and documents with team members through team-based change-management practices referred to as software configuration management (SCM).

SCM best practices have been codified into a series of SCM patterns that describe effective change-management practices for team-based software development. Organizations that are heavily involved in software development follow such best practices to ensure that they implement repeatable processes for creating high-quality software. The Software Engineering Institute (SEI) Software Capability Maturity Model (SW-CMM) is an example of enterprise-level practices designed to improve software quality and productivity. (Change management is a significant part of the SW-CMM, but the model describes other best practices as well.) Distributed development across multiple organizations pushes the limits of SCM and the change-management elements of SW-CCM.

The first part of this chapter sets the stage for software development, beginning with a discussion of the software development lifecycle, methodologies supporting that lifecycle (such as waterfall, spiral, rapid application development—RAD—and extreme programming), and the role of SCM. The focus then turns to change-management issues at the enterprise level and best practices such as the SW-CCM. The final section of the chapter discusses limitations of silo-based SCM and the role of ECM in addressing these issues.

Understanding the Software Development Lifecycle

The software development lifecycle consists of six stages (see Figure 3.1):

1. Requirements analysis
2. Design
3. Development
4. Implementation
5. Maintenance
6. Retirement

A common trait among each of these stages is the need to manage change and unanticipated consequences. In some cases, the need for change can appear early in the stage. Problems during the implementation stage are often apparent soon after the process begins. However, design flaws are often not apparent until much later stages, when the system is in production. Change management in each stage of the software development lifecycle is essential to creating quality software that meets requirements and functions properly in the IT environment. The first step in realizing that goal is to analyze requirements.

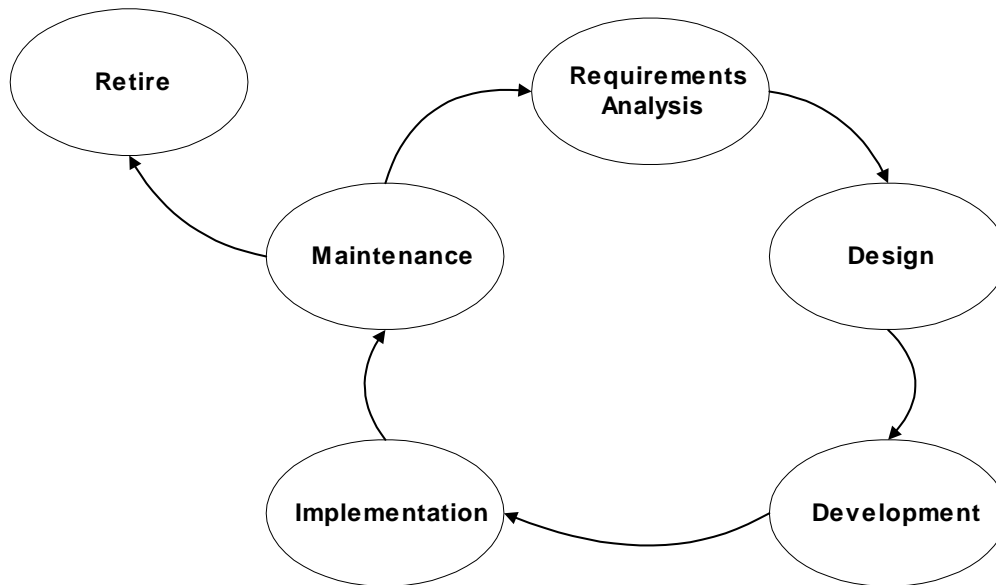


Figure 3.1: Software development across industries and application types follow a common lifecycle.

Analyzing Requirements

Software development begins with a need. In some cases, the need is broad and extensive: banks and healthcare institutions need to comply with new privacy regulations, and financial services companies need to comply with the Sarbanes-Oxley Act on financial reporting. Other requirements are more focused: the marketing department of an insurance company needs to understand which agents and brokers are most effectively enrolling new customers, and a virtual team needs a Web-based content management system to help its members collaborate. Understanding the nature and scope of a business need is the first stage of the software development lifecycle.

Gathering requirements is challenging. IT analysts work with users familiar with the business need, often referred to as subject matter experts (SMEs), to build a bridge between the business and technical realms. The result of this collaboration is a requirements document. A challenge faced by IT analysts and SMEs is understanding when they have enough detail. On one hand, too little attention paid to gathering requirements leads to applications that do not effectively solve a problem. On the other hand, too much focus on requirements gathering leads to “analysis paralysis.” Clearly a balance is required. Requirements must be clear and concise as well as precise enough to enable design and development. The scope of requirements should not be so broad that gathering precise and accurate requirements hinders the progress of a development effort. Spiral, RAD, and extreme programming methodologies have evolved to support the solicitation of quality requirements within the resource constraints of most projects.

Change Management in the Requirements Analysis Process

Change management is required in the analysis stage to control requirements documentation: As multiple analysts and SMEs contribute to requirements gathering efforts, change-management policies are necessary to define how requirements are updated. Workflows define how content is checked in to repositories, how older versions of documents are tracked, and who is responsible for reviewing and prioritizing requirements.

When questions about requirements or conflicts between requirements are identified in later stages of the lifecycle, they often introduce changes to requirements documents. Ideally, designers and developers could move from one stage of the lifecycle to the next with confidence that earlier phases are complete. However, such is rarely the case. During each stage, designers and users learn more about the system under development and its role in the organization. These insights can change decisions made in earlier stages. Fortunately, change-management practices track information and assets, making it easier to adjust earlier work than if designers used an ad hoc process.

Designing Software

The design phase of the software development lifecycle begins when analysts have sufficient understanding of the requirements. During this stage, software designers and architects create:

- Process models—Define the logical rules for how data flows through the application.
- Data models—Show how data is organized within databases (physical data models illustrate how data is stored).
- Logical models—Describe how pieces of information relate.

Models of application logic illustrate how processes are implemented and data is managed. Programmers use these models like blueprints to develop the code that actually implements the design. As Figure 3.2 shows, process models, data models, and application logic are dependent upon each other and upon the requirements gathering stage.

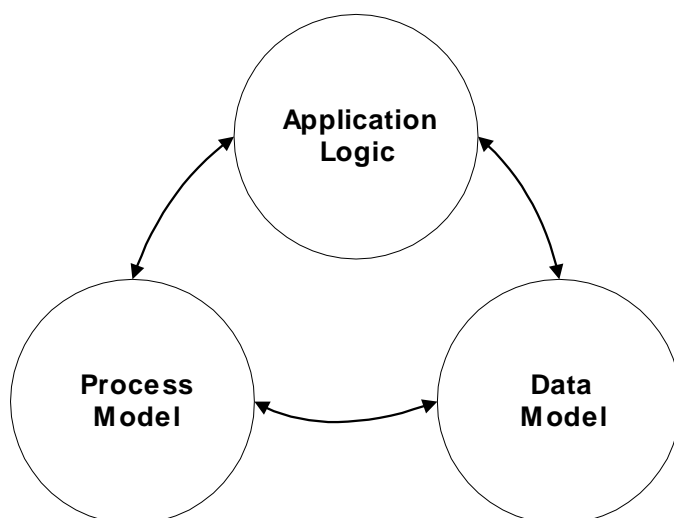


Figure 3.2: Design components are interdependent—changing one affects the others.

Developing Software

Software developers were some of the earliest adopters of change-management practices. In the late 1960s, NATO and the United States military promoted SCM to improve software quality and development practices. Today, change-management tools are as common as editors, debuggers, and other programming utilities. These tools are essential given the complexity of the software development lifecycle.

In the simplest case, a single programmer receives a design document, develops a program, tests it, corrects errors, and turns the program over to users. Let's assume that the design remains the same during the course of development (a rare occurrence). The program is divided into modules, and the programmer develops one module at a time, moving on to the next module after the previous one is completed and tested. Ideally, when modules are tested together they continue to work as expected (another rare occurrence). Errors, unanticipated results, and questions about design specifications are resolved at this point. Programmers then "lock" working and nearly working pieces of code before adding new features or correcting errors. The locked code is a *version* in SCM terminology. As Figure 3.3 shows, programming has a lifecycle within the broader software development lifecycle.

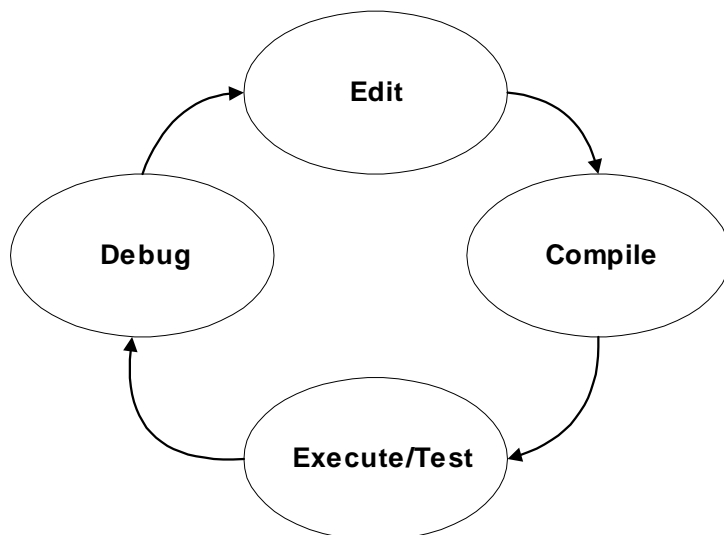


Figure 3.3: Programming is an iterative process and version control is essential to preserving development work.

When programmers integrate modules developed by others into existing systems, change management becomes even more complex. In such cases, programmers need to coordinate with the developers of the modules to understand:

- The behavior of modules developed by others
- Protocols used to share data between modules
- How to adapt programs to the modules or vice versa

The need for change management is also evident in the process to develop multiple versions of an application to work with different OSs. Client/server applications, for example, might have a Microsoft Windows, Apple Macintosh, and UNIX version.

Adding to the complexity of the development phase, some developers within an organization might be working on application maintenance on production systems while others are working on the next version of the application. Thus, for complex systems under constant development and use, the application is frequently being reworked and then re-implemented. Change-management practices are necessary for choreographing these processes and managing the lifecycle of such applications.

Implementing Software

Moving software from development to production requires clear understanding of the new program and the environment in which it will operate. Developers often test programs on a variety of platforms during the development phase, but unanticipated problems can still occur during implementation. For example, the implementation of a new PC-based application might disrupt existing programs, and programs that generate a great deal of network traffic might run too slowly over dial-up lines. Although a best practice is to test for as many of these scenarios as possible, unanticipated incidents inevitably occur.

Thus, developers and users need to coordinate installations with the administrators who are responsible for servers and applications affected by the installation. In large organizations, this task is demanding enough to warrant the specialized role of a release engineer.

The release engineer's primary responsibilities include accommodating the needs of the new system, ensuring that existing systems continue to function during the implementation, and coordinating with both IT and business stakeholders. To manage essential information for each of these responsibilities, release engineers use change management systems to:

- Design documents, such as process models and physical data models (these depict workflows, integrated systems, and the location and volume of data).
- Control versions of information created during the development stage, including details about application versions and platforms, source code, configuration files, data models, process models, and other software-related assets.

Maintaining Software

Software requires two types of maintenance. Minor changes are often required to revise features and correct errors. Slight changes to the interface to improve contrast between colors or re-ordering menu items are maintenance changes. Fixes to minor bugs, such as an incorrect link in a Web application or incorrectly formatted data generally constitute maintenance. The second type of maintenance is infrastructure maintenance. Examples of this type include backing up data and rearranging database storage schemes to optimize performance.



Maintenance is often the longest phase of the software development lifecycle.

When entirely new features are added to a system or major modules are redesigned to improve performance, the application has moved out of the maintenance phase to the revision and improvement phase, which is a return to the beginning of the software development lifecycle. At this point, developers will need to work with SMEs to gather new requirements for a redesign of the software. The implications of this distinction are realized clearly in change-management practices: During the requirements analysis and design stages, developers revise and create documents that are subject to strict change management. Plans are created for testing the new features and implementing the new code in production. These test plans, in turn, are dependent on test plans and scripts developed during the requirements analysis and design stages. Although maintenance procedures are documented and tracked, the process is not as detailed as that of the requirements gathering and design stages.

Maintenance operations are triggered by change requests. These requests often involve the need for a new feature or bug fix. There are no rules for determining when a feature request should be implemented as part of maintenance or held off until a new development cycle. In general, isolated changes, such as adding a menu item to the user interface, are accommodated in maintenance cycles. Changes that entail multiple modules and data sources often require extensive analysis and are best handled during a full development cycle. Change requests follow a well-defined process (see Figure 3.4).

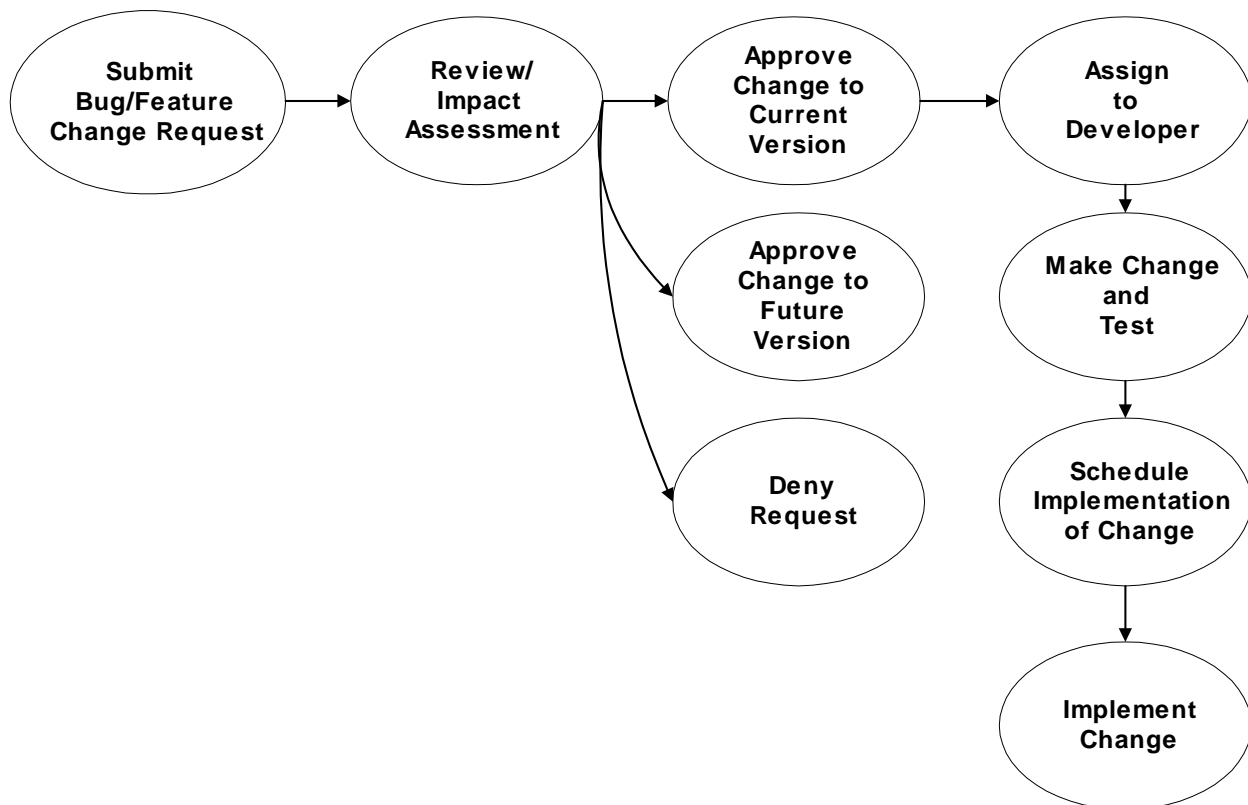


Figure 3.4: Change requests initiate a workflow designed to correct errors and address minor feature changes.

Retiring Software Applications

Eventually, an application is no longer viable in an environment. This occurs for many reasons:

- Business operations change.
- Better software options become available.
- Vendors no longer support older systems.
- Organizations consolidate IT systems with enterprise applications such as enterprise resource planning (ERP) and customer relationship management (CRM) systems.
- The cost of maintaining and operating an application is not justified, particularly if the application requires a proprietary OS or is supported only on older hardware.

The retirement of such applications often occurs while introducing a new system. For this process, change-management controls are the focus of migration efforts. Migration teams need to address several basic issues:

- Will users stop using the retiring system and switch to the new system immediately or will the two run in parallel?
- If the two systems run in parallel, will they perform duplicate tasks or will tasks be divided between them?
- How will data be synchronized between the two systems running in parallel?
- How will data migrate from the retired system to the new application?
- How will the transition affect other applications that integrate with the retiring system?
- What business operations are affected during the transition?
- What is the backup plan if there are problems with the transition?

To answer these questions, the migration team needs information about dependencies, workflows, and roles—the type of information maintained in ECM systems. As Figure 3.5 shows, change-management information is generated and used at every stage of the software development lifecycle.

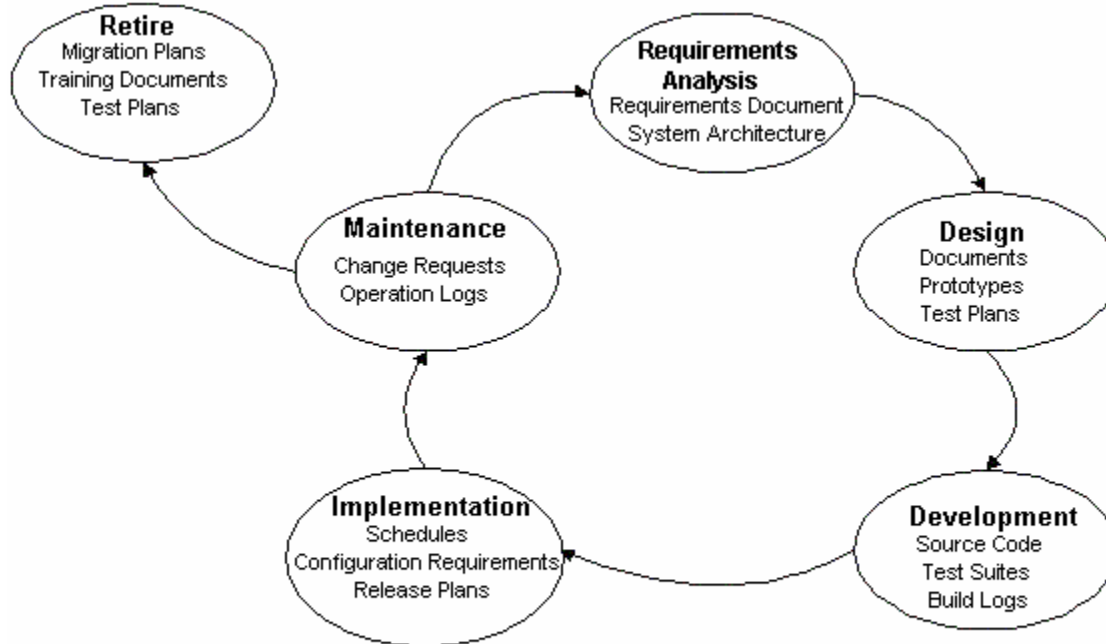


Figure 3.5: Change management is a fundamental process in every stage of the software development lifecycle.

Most developers agree on the basic stages of the software development lifecycle, but there are differing views on the best way to execute those stages. The requirements analysis, design, and development stages generate the most debate and the result is the existence of several methodologies for software development.

Choosing a Methodology for Software Development

The four software development methodologies in common use are:

- Waterfall
- Spiral
- RAD
- Extreme programming

Each methodology has benefits and drawbacks. They vary in the emphasis they place on separating each stage of the lifecycle and the level of demand for documentation as well as other facets controlled by change-management systems.

Waterfall Methodology

The waterfall methodology is the oldest of the four. Its basic principal is that once a stage has been finished, you do not return to that stage, as Figure 3.6 shows.

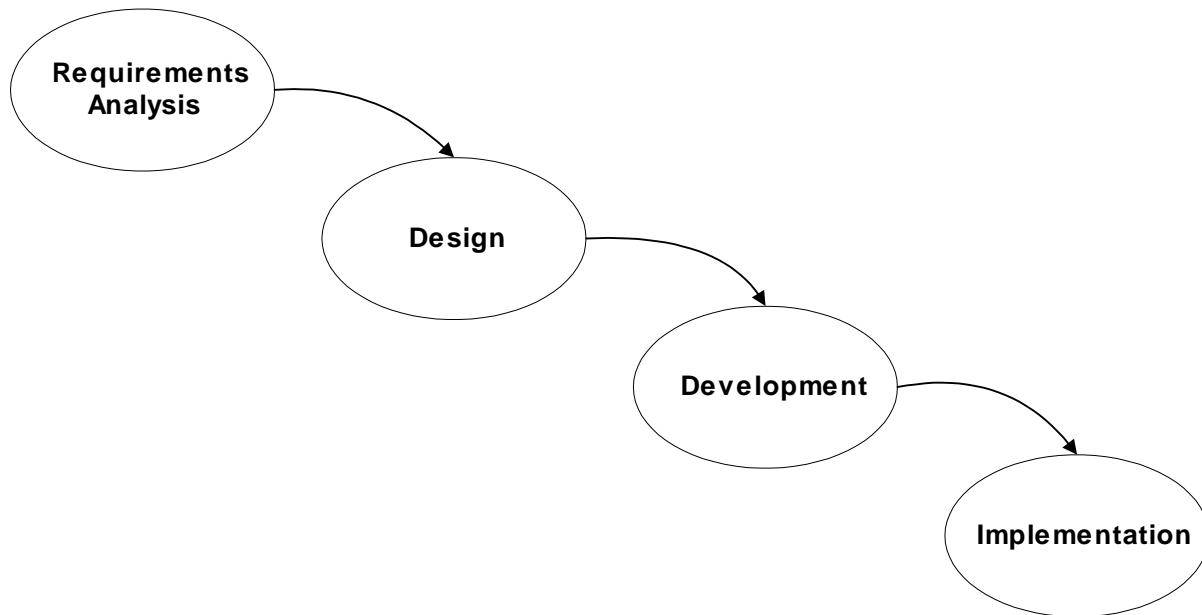


Figure 3.6: Once you pass a stage in the waterfall methodology, you do not go back.

The maintenance and retirement phases are outside the scope of this methodology. The next iteration of the process, which begins a new requirements analysis phase, is considered a new project.

This approach forces stakeholders to define all of the business objects of the system before starting the design, finishing the design before developing code, and completing the code and testing before implementation. The advantage of this approach is that objectives are clearly defined early in the process. Changes are less expensive to implement in the requirement stages than in the development stage.

Advantages of the Waterfall Methodology

With a clear understanding of requirements, designers are more likely to create a robust application during the design stage. Design changes introduced during the development stage are often less-than-ideal implementations.

Consider an online order processing system. If designers know that a series of frequently changing business rules are required to validate an order, the designers can accommodate this need with a flexible module for defining and editing business rules. If those requirements for validation rules are not discovered until the development stage, programmers might have to resort to a quick fix such as putting the business rules directly into the program. In such a case, the program would need to be changed each time the business rules change. As the number of business rules increases, it becomes a more and more difficult task to implement quick fixes that do not interfere with other parts of the program. Avoidance of this type of implementation is a key driver behind the use of the waterfall methodology.

Disadvantages of the Waterfall Methodology

The most significant disadvantage of the waterfall methodology is the difficulty in making changes to requirements and designs after those stages have passed. For example, if developers follow the waterfall methodology in the strictest sense, they would not accommodate a requirement discovered in the late design stage or early development stage. Clearly, in dynamic business environments where needs are constantly in flux, this rigidity outweighs the benefits of the methodology.

Another limitation of the waterfall methodology is that the marginal cost of identifying requirements and making design decisions can increase as the duration of these respective stages grows. Consider the following example.

It is fairly easy to elicit requirements for basic user interface functionality. Users can describe the tasks they perform and the functionality they expect based on their past experiences. Less clear is how a new application will *change* the way users work. Once a final application is in place, users might experience a significantly different daily working environment. The requirements that the end users identified during the requirements analysis stage were based on work patterns that might no longer exist.

Analysts and SMEs using a waterfall methodology might extend the analysis phase in an attempt to model every anticipated change to users' work patterns as a result of the new program's implementation; however, there is no guarantee developers could accurately predict subtle changes in work patterns once the new application is implemented. In many cases, it is simply less expensive and faster to build a system, let users work with it, then solicit feedback that may result in additional requirements. This system is the basic idea behind the spiral methodology.

Spiral Methodology

Using the spiral methodology, designers work in short phases to gather requirements, create a design, develop code, and implement and evaluate the application. The series of short phases is repeated, and each iteration adds more functionality to adapt to changing requirements. Figure 3.7 depicts the spiral methodology process.

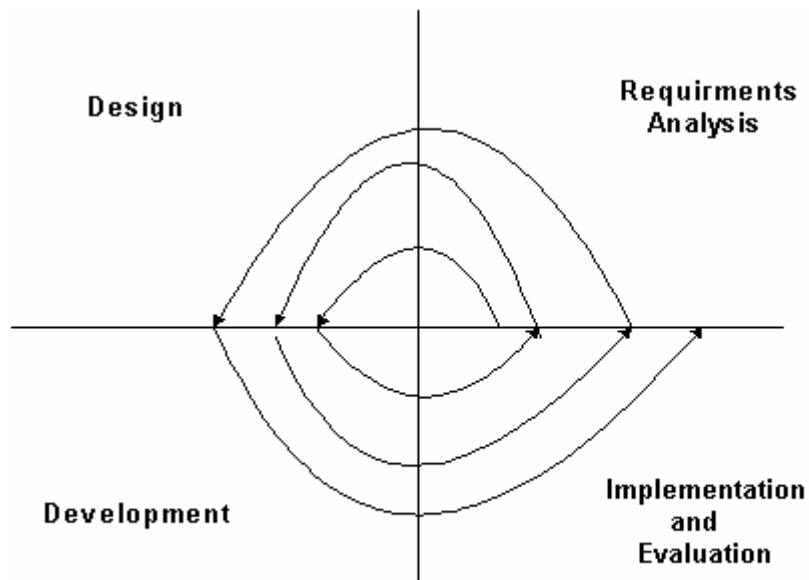


Figure 3.7: With the spiral methodology, each phase is repeated several times.

Advantages of the Spiral Methodology

The spiral methodology is especially adept at accommodating change. Requirements discovered during the development or implementation phase do not cause problems as they do in the waterfall methodology. Because each phase is short and eventually followed by additional requirements analysis and design stages, the newly discovered requirements are readily accommodated.

The incremental deployment of the spiral methodology allows users to work with portions of the newly developed system early in the development process and provide valuable feedback to the design team. This information is especially useful when business processes change in response to the new system. Business users do not need to anticipate all possible changes in their work patterns early in the development effort.

Disadvantages of the Spiral Methodology

A drawback of the spiral methodology is that it does not define a fixed number of cycles, which can lead to undisciplined requirements gathering—if there is an error or omission, developers might determine they can address it in a later cycle. In addition, unnecessary features might be added and eventually discarded. Without a fixed design, it is difficult to fit the system into the broader IT infrastructure—as applications become more integrated, particularly with the advent of Web Services, frequent changes to applications cause ripple effects.

RAD

RAD is another methodology that uses short lifecycle stages, but executes each stage only once. The goal of RAD is to develop applications that meet basic business requirements while keeping to a short schedule.

The RAD methodology uses requirements gathering, design, development, and implementation stages, but they are far less formal than in the waterfall or spiral methodologies:

- Requirement and design sessions are done rapidly with end users and other stakeholders.
- Meeting notes and prototypes built on the fly substitute for formal requirements and design documentation.
- Code libraries and third-party modules are used whenever possible to minimize development time, even if quality or functionality suffers.
- Set amounts of time are allowed for particular tasks, such as implementing a user interface; tasks that cannot be performed within that time are dropped. This process is known as *time boxing*.

Advantages of the RAD Methodology

RAD's focus on time boxing helps keep projects on schedule. Users know they will have something to work with at a certain date. The program might not contain all the features they expect, but that is a tradeoff of an exact delivery date.

Designers and developers can change the application design at almost any time. As new requirements are found, the requirements can be accommodated without waiting for another cycle of the spiral methodology or navigating ad hoc change request processes of the waterfall methodology.

The development team receives constant feedback in a RAD project. End users and other stakeholders meet frequently with developers to make design decisions, test the application, and provide direction to the project.

Disadvantages of the RAD Methodology

RAD will not work with all software development. Applications developed with RAD should have a fairly small number of users, require little integration with other systems, and have few performance requirements.

For RAD development projects, developers do not create formal documentation, which makes project management a difficult task. Progress is hard to track. There are no peer-review processes, so stakeholders might find it difficult to get the information they need. Development teams can isolate themselves, losing sight of architectural constraints imposed by the IT infrastructure. Change management is essentially impossible with RAD projects.

ECM has evolved on the principals of dependencies between organizational assets. IT systems, strategic plans, business processes, and LOB operations are interrelated. RAD implicitly assumes project isolation from these assets and processes and is basically evolution in a vacuum, so it is best suited for research and development arenas.


Extreme Programming

Extreme programming is one of the newest software engineering methodologies and builds on many of the practices developed in earlier methodologies. Extreme programming development centers around teams comprised of a customer, analysts, designers, and programmers. The customer defines the business objectives, prioritizes requirements, and generally guides the project. Designers and analysts work with the customer to translate the business objectives into simple designs. Programmers work to code the design.

In this methodology, roles are less formal than in other approaches. Designers may program and programmers might work with customers to define requirements. Extreme programming teams do not try to map out the entire project. Instead, they focus on two questions:

- What to do next?
- When will it be done?

This practice provides flexibility similar to that of the RAD and the spiral methodologies. Projects that use extreme programming focus on short-term deliverables, such as delivering a new version of an application every 2 weeks. The regular delivery schedule prevents long delays and gives customers the ability to prioritize and identify requirements throughout the development effort.

 For more information about extreme programming, see <http://www.xprogramming.com>.

Advantages of the Extreme Programming Methodology

Extreme programming places emphasis on quality. Teams build applications by making small additions and improvements and testing with each change. The goal of this technique is to always improve a program without introducing bugs. Code is continually improved—in addition to adding new functionality in each release, programmers revise existing code to maintain quality standards. This methodology is designed for parallel development. Pairs of programmers work on separate modules but integrate and test them regularly.

Disadvantages of the Extreme Programming Methodology

Extreme programming works well for small and mid-sized projects, but is not suitable for large development efforts. The focus of close coordination between team members cannot scale to teams of hundreds or more. In addition, this methodology assumes that developers work with a customer with full authority to make design decisions. Some decisions require outside approval from either business or technical areas of the organization. Outside decision makers play a greater role as the level of integration with other applications increases.

Methodologies and ECM

Assets, processes, and dependencies between assets and processes are the building blocks of ECM. The major software development methodologies manage these building blocks to varying degrees. The fact that there are at least four major methodologies for dealing with the well-defined and agreed upon software development lifecycle attests to the difficulty of balancing the need for controlled structures with the need for flexibility in development efforts.

The waterfall, spiral, and extreme programming methodologies are all amenable to change management. RAD does not fit well with ECM practices. Although it is a useful methodology for small, non-mission critical, low-risk projects that focus on the need to deliver in a short period of time, RAD projects tend to generate little documentation, have few quality review checks, and produce few formal deliverables.

To implement change management in the software development lifecycle, teams must follow formal processes and create standard assets such as:

- Requirements documents
- Design documents
- System architecture documents
- Test plans
- Test results
- Programming code and executable programs
- Change request documents

Documents and code are often managed through different systems, but the two systems should be kept closely linked. Versions of a program should be associated with a particular version of a design document, which, in turn, is associated to a particular version of the requirements document. Test plans and test results should be associated with corresponding versions of an application. Coordinating documents and code in different repositories is one of the significant challenges of silo-based change-management practices.

The Need for Traceability

Change management provides for traceability. Changes in code are associated with a particular version of the code, are tied to specific requirements or change requests, and generate test results. Information about a project—both its design goals and the history of its development—should be available to team members and other stakeholders.

Organizations choose methodologies based upon their need for traceability, which is dependent on factors such as:


- Size of the organization
- Level of integration among projects
- Size of projects
- Duration of projects
- Lifespan of the software under development

Large organizations, large projects, and projects that have long development cycles require more change management and traceability than smaller efforts.

Transparency is essential for proactively managing change. For example, consider a Web Service developed by a RAD team to allow a few supply distributors to check the status of their accounts. The program is a success, and management decides to roll out the same service to individual customers. Questions arise immediately:

- Will the service scale to meet the number of expected users?
- How is security managed?
- How extensive was the testing?
- How will the additional load impact the current architecture?

If the team did not formally document its processes, management has to depend on the memory of team members, assuming they are still with the company.

 Poor documentation and lack of controls affects projects and operations well beyond the application under development. In addition, government regulations such as the Sarbanes-Oxley Act require the maintenance of documentation.

Large and mission-critical projects require extensive change-management support for a number of processes. Change-management systems must support auditing to enable users and developers to ensure that all functional requirements are met by comparing requirements with the final system. For example, if a requirement is not implemented in the system, there should be documentation describing when and why the requirement was dropped and who authorized its removal. Three types of audits are conducted in large projects:

- Functional configuration audits document that tests are conducted according to test plans.
- Developers and implementation teams use physical configuration audits to compare software components with design specifications.
- Configuration verification processes match change requests to configuration items and versions to ensure post-design changes are implemented.

Demands for Traceability Beyond Software Development

The tools and documentation that make up ECM procedures serve more than just software development projects. Executives and managers cannot make decisions and execute plans without information about the impact of change. Responding to unanticipated consequences of change is costly and inefficient. Software engineers first turned to SCM to improve their own development processes. The need for those practices now extends beyond single development projects into the broader organizational realm (see Figure 3.8).

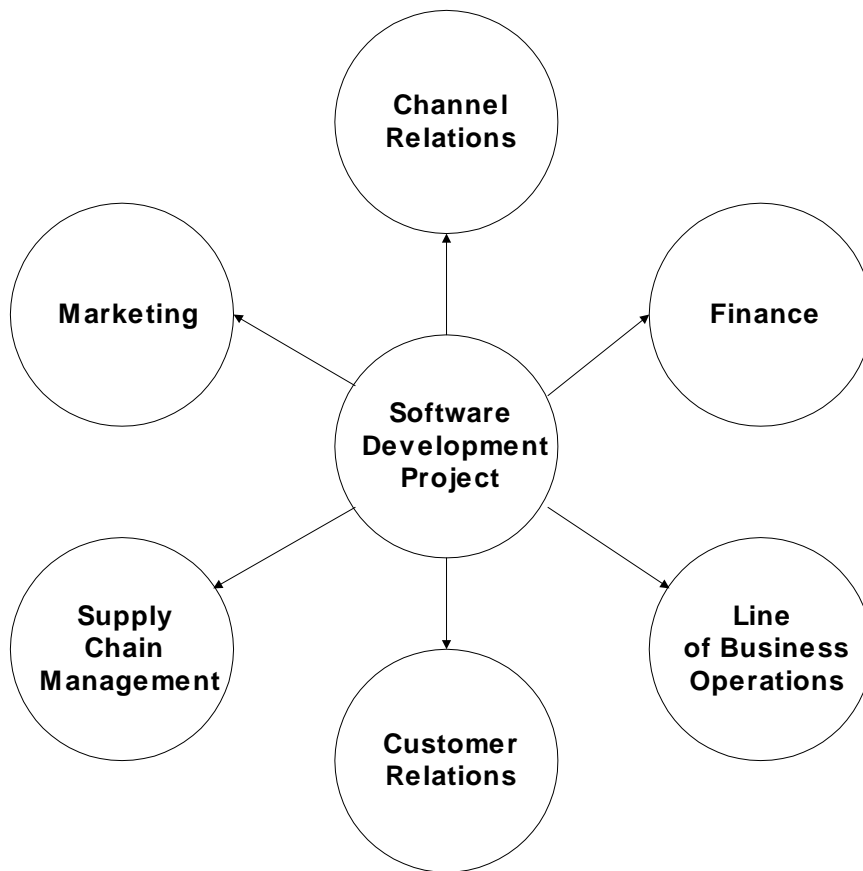


Figure 3.8: Changes in software development projects can cause changes throughout an organization.

In the past decade, practitioners have realized the need to formalize software development practices at an organizational level. The most well-known example of this approach is the SW-CMM.

Supporting Software Change Management Across the Organization

The military was the initial driver behind the move to improve software development across projects and organizations. The United States government employs large numbers of contractors to develop applications for the military, thus the government required a method for assessing the ability of contractors to deliver high-quality software that meets complex requirements. The SW-CMM is the basis for one method to rate contractors. More important, it offers a collection of best practices for controlling software development across large organizations.

The SW-CMM

The SW-CMM describes practices that move software development from ad hoc procedures to disciplined repeatable processes. The model consists of five stages:


1. Initial
2. Repeatable
3. Defined
4. Managed
5. Optimized


During the initial stage of maturity, an organization lacks well-defined processes. Software development is ad hoc and sometimes chaotic. No formal project management or change-management controls are in place.

In stage 2, organizations use management procedures to track project schedules, costs, and functionality. Software configuration practices, such as version control systems and policies for development and release management, are introduced at this stage. Although not fully standardized across the organization, these practices are repeatable across similar projects.

By stage 3, organizations have developed standard software development practices that monitor and control project administration and software engineering. In the defined stage, organizations are managing software development at the enterprise level rather than on a project-by-project basis.

In stage 4, organizations maintain detailed metrics on software process and product quality. Projects are managed using these quantitative measures. The final stage, optimized, addresses continuous improvement through quantitative feedback mechanisms.

 SEI is currently creating an integrated maturity model called the Capability Maturity Model Integration (CMMI) that includes maturity models for software, systems engineering, and integrated product and process development. This move reflects the trend in change management away from silo-based change-management systems to ECM. The goal of CMMI is to eliminate redundancies between maturity models, improve efficiencies, and make the model more applicable to small and mid-sized projects. Eventually, CMMI will replace separate maturity models; however, its development is still in the early stages. The discussion of the SW-CMM in this chapter focuses on elements from the SW-CMM that will likely remain in the CMMI.

 For more information about the CMMI, see <http://www.sei.cmu.edu/cmml/cmml.html>.

The initial stage of the SW-CMM does not entail change-management control, and the later stages assume that change management is already in place. Thus, the following discussion focuses on change-management implementation in stages 2 and 3.

The SW-CMM and ECM

The SW-CMM describes what organizations do at various maturity levels but does not describe how to accomplish these tasks. Let's explore specific change-management processes in use at the project and organizational levels.

Managing Change Within Software Development Projects

Change management is well understood at the project level. The core assets managed are:

- Workspaces
- Code lines
- Builds
- Test suites
- Baselines
- Releases

These components are the basic tools and products of software development.

Workspaces are areas in which individual programmers develop and test their code. When a module is finished and ready for use by others, it is checked in to a version control repository. When it is time to work on that module again, it is checked out so that others on the team know it is being modified.

Version control systems prevent or at least warn developers when two or more users have checked out a file, thus preventing one programmer from overwriting the changes of another programmer. Programmers check in their code as soon as possible, making it available for others to use. If code is not checked in frequently, the module becomes frozen, and others are forced to use older versions of the module for their development and testing work. Checking in also allows the programmer to incorporate the latest code into the code line.

Code lines are sets of files and other components that comprise a software product. A common practice is to maintain a main code line that represents the core of the product. Branches split from the main code line to create a working set of files for new releases. For example, a development team releases version 1 of a program to users. Several designers and programmers begin working on version 2 to add more features. A short time later, a user finds a bug in version 1 and requests a change. How is the problem corrected?

One option is to simply change both versions 1 and 2 and continue with development. In simple cases, this method might work, but it is not a viable solution for most projects. A better option is to have both versions of code branch from the same main line, make the change in version 1, propagate the change to the main line, then update version 2 with changes to the main line. Figure 3.9 illustrates this process.

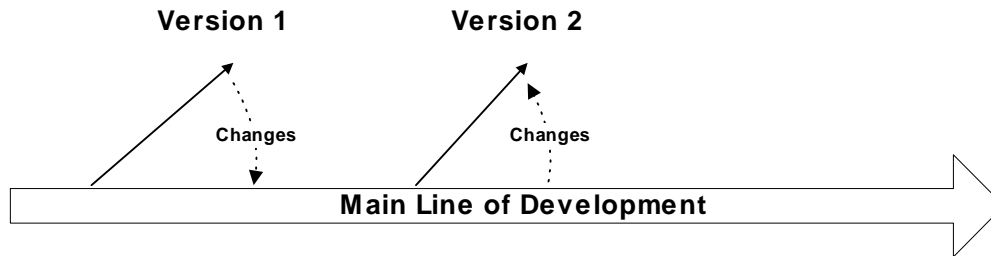


Figure 3.9: With main line development, changes in one version are saved to the main line of code and propagated to other versions.

A change-management policy is associated with each line of development that describes the purpose of the code line, who can make changes, how frequently code is checked in, and other controls on the development process.

Once programmers have written code, it must be combined and linked with supporting programs, such as third-party components and module libraries, and translated into an executable program. This process, known as a *build*, is a basic operation subject to a policy. Builds should be done frequently to enable developers and test engineers to test the complete system and identify bugs soon after they are introduced. Builds should generate log files with information about components that were included in the build, which tools were used to create the build, and any errors that occurred. These log files are an integral part of the project change management.

Test suites are collections of tests that exercise the functionality of a system. Tests evolve along with the development of the software and are subject to similar change-management procedures as those used to manage program code.

A baseline captures and records assets and asset configurations based on selection criteria. In addition, a baseline ensures that such configurations and assets are frozen, facilitating rollback if necessary as well as recording project milestones and deliverables.

Software is rolled out to users through releases. A release consists of a version of the system that has passed build and test processes and is ready for use. Change management includes policies that describe the intended audience for the system, release notes and other documentation, schedules, and plans for coordinating with systems administrators. Figure 3.10 illustrates the central nature of a change-management repository within the SCM process.

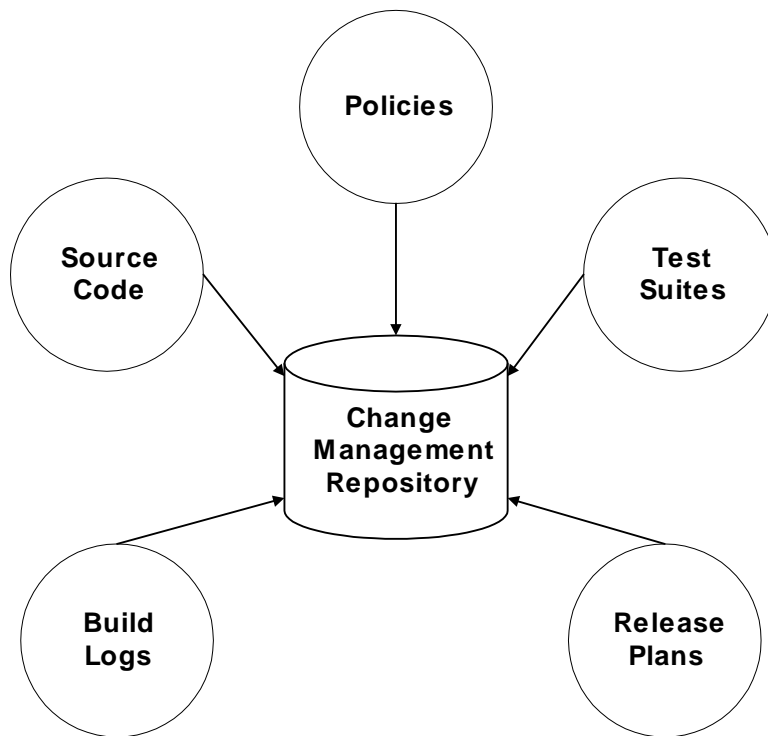



Figure 3.10: SCM requires a centralized repository for different types of assets.

Common processes for managing software development assets are emerging among developers. One set of processes are known as SCM patterns. These patterns describe typical operations and best practices for performing them.

 Brad Appleton and Stephen P. Berczuk's *Software Configuration Management Patterns: Effective Teamwork, Practical Integration* (Addison-Wesley) describes 16 core patterns for effective SCM. There is also information about SCM patterns at <http://groups.yahoo.com/group/scm-patterns/>.

Using SCM Patterns

SCM patterns describe the context and processes associated with managing software development assets. (Patterns originated in software engineering as a means of capturing commonly used programming techniques.) Four categories of SCM patterns include:

- Organizational patterns
- Architectural patterns
- Process defining patterns
- Maintenance patterns

Organizational patterns describe how teams are organized and managed. Architectural patterns specify how to structure software at a high level. Process defining structures describe how to establish workspaces, directory hierarchies, and other supporting systems. Maintenance patterns cover common operational patterns of the team.

Patterns consist of a general rule governing a process and a context for which it is applied. They are formally defined by:

- Context
- Problem
- Forces
- Solution
- Variants

The context describes at which point in the software development lifecycle the pattern should be applied. Deciding how to structure code lines or define a policy for checking in code are examples of contexts.

The problem describes the issues that need to be resolved, such as when changes in versions should be merged into the main line and to which code line should a developer save a change.

Forces are factors that influence how a problem is solved. For example, the length of time a module can remain checked out is dependent upon how that frozen code affects the development of other modules.

Solutions are best practices that take into account forces and still solve the pattern's problem. For example, before merging code into the main line, the code must be thoroughly tested. If the test suite for the module requires more than 1 hour, consider not merging more than once a day.

Variants describe differences in patterns that are closely related. For example, the Merge Early and Often pattern described at <http://www.cmcrossroads.com/bradapp/acme/branching/branch-policy.html#MergeEarlyAndOften> has two variations. One addresses how changes already merged into one code line should be merged into another code line; the other describes how to manage high-volume merges.

 For a thorough discussion of SCM patterns for controlling branching in code lines and several fully described patterns, see <http://www.cmcrossroads.com/bradapp/acme/branching/streamed-lines.html>.

Developing Policies and Processes for SCM

Managing change in software development is a multi-level and multi-dimensional task. Software development follows a well-defined lifecycle, but the processes that move projects through that lifecycle vary depending upon the methodology used. Methodologies also dictate aspects of change management, such as the amount of documentation created, the formality of process reviews, and the degree of control imposed on developer's code management. Effective management of such complex environments necessitates policies and procedures.

Version Control Policies

Version control policies dictate how programmers add code to and use code from the version control repository. They describe criteria for checking in code, checking out code, merging changes, and other basic code-management operations. Understanding how these operations are performed allows project teams to develop subsidiary policies, such as access control and build and release policies.

Access Control Policies

Access control policies define who can change particular modules within a source control system. These policies establish who can update specific branches of a code line.

Build, Baseline, and Release Policies

Build, baseline, and release policies describe how code is managed in a version control system, collected from the repository, and compiled into an executable application. Build policies should consider the frequency of builds, the types of tests run after the system is built, the level of detail logged during the build process, and which outputs from the build process are placed under version control. Baseline policies describe which team members have the authority to create a new baseline and under what circumstances. Release policies address when and how code moves from development through various stages of testing and finally into production.

Additional Policies

The software development process, like so many business processes, cannot be isolated to a single domain. Changes to assets and processes within the software development process affect other business operations. Similarly, changes outside of development efforts can have significant impact on those efforts. Policies related to non-software dependencies include those that address:

- Enterprise architecture restrictions
- Network utilization
- Changes in requirements (methodologies dictate how these are handled)

Development of integrated systems requires coordination with others. Many integration issues—for example, data exchange protocols, error handling procedures, and performance commitments—are best addressed during the design stage. Overall guidelines for testing integrated systems are best defined in policies at the start of a project. These can delimit, for example, boundaries for testing (especially if some testing requires production systems) and the role of systems administrators in the development project.

Summary

Software development is a dynamic process that touches many points of an organization. Change management is a factor of the software development lifecycle from several perspectives. At the individual level, change management enables developers to build and test modules while maintaining fall-back versions of code that is known to work. At the project and team levels, change management enables parallel development of multiple versions of software by a range of developers using several methodologies. Enterprise-scale software development requires more than silo-based change management. ECM practices are key constituents of mature software development processes.

Chapter 4: Managing Change in System Configurations

IT infrastructure is a lot like an orchestra—they comprise a wide range of instruments that act in concert to create a unified product, be it music or information flow. When the instruments work together, the results are favorable; however, a slight variation in even one instrument—a wrong note from the French horn or a misconfigured router table—can skew the overall operation of the group. Managing system configurations is like conducting an orchestra—it takes knowledge of each instrument’s role in the overall process, the ability to identify variations from the ideal, and a clear vision of the final product of your efforts.

This chapter focuses on managing the hardware, software, and network components that are the foundation of an IT infrastructure. We’ll explore problematic system configurations that have consequences that affect more than just network services. I’ll provide examples that demonstrate the subtle dependencies between system configurations and business operations. Next, the chapter outlines the goals of system configuration management followed by a discussion of configuration management and the ECM model outlined in Chapter 2. Finally, I’ll briefly discuss challenges particular to system configuration management.

Interdependence in System Configurations

An IT infrastructure is made up of a wide range of devices that perform a variety of functions and create many interdependencies. The effects of such interdependencies can be obvious—large-scale failures or misconfigurations such as an offline database server or an overtaxed email server—or less easily identified. It is much more difficult to identify problems with smaller or embedded components although the results in many instances are just as detrimental as large-scale problems.

For example, consider a business intelligence application that uses a moderately sized data warehouse. Executives, LOB managers, and project managers depend on this data warehouse for reporting on key performance indicators, operational measures, and ad hoc querying. Users access the data warehouse through a portal and a Web-based interface to the database (see Figure 4.1). As long as the portal server, the database server, and the network are operating, the system is functioning, right? Not necessarily.

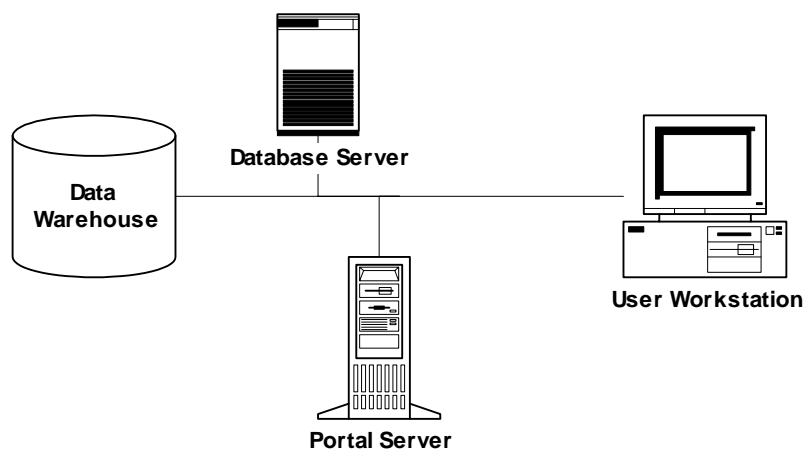



Figure 4.1: A small business intelligence system looks simple at a high level, but this figure hides the complexity of the underlying infrastructure.

Understanding Implementation Details

Let's take a closer look at the components. The device labeled data warehouse stores the information managed by the database server. In more detailed terms, the data warehouse is a storage area network (SAN)—an array of disks configured to act as a single logical storage device.

 SANs have many advantages for network administrators. They allow multiple servers to share disk storage, reducing the chance of wasted space on individual servers. A single SAN requires less administration than several separate server-based storage systems.

How SANs Fit in the Infrastructure

In a typical SAN environment, multiple servers will use a single SAN for storage, as Figure 4.2 shows.

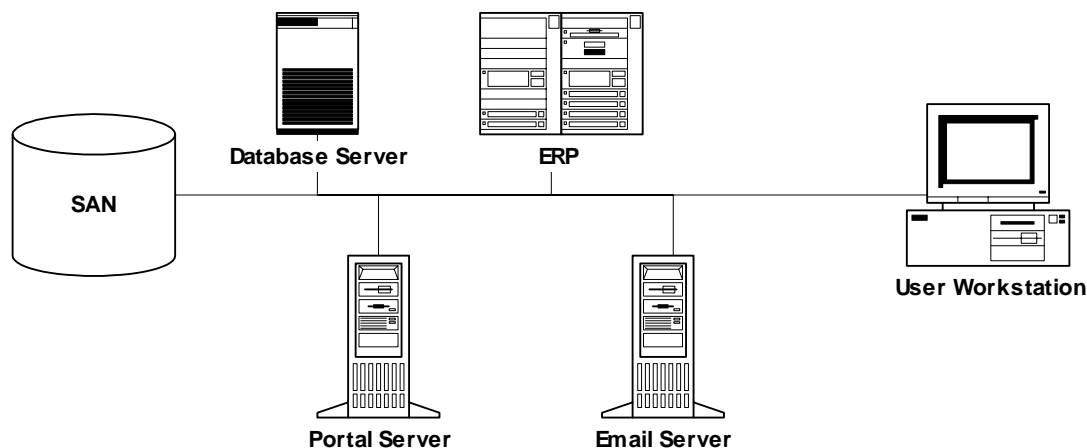


Figure 4.2: Multiple servers often depend upon a single SAN.

Data moves in and out of the SAN through channels that respond to requests for data from controllers found on servers. Channels are like traffic cops, they decide which data moves and how the data moves. When a server makes a request for data, the channel determines how much bandwidth to allocate to that server based on the overall demand on the SAN. For example, if three servers are pulling data from the SAN, the controller might allocate 25 percent of the controller's bandwidth to the fourth server. If only a single server is retrieving data at one time, the controller might allocate 50 percent of the controller's bandwidth and keep 50 percent in reserve in case another server makes a request. If, after some time, no other server makes a request, the controller may decide to allocate 100 percent of the bandwidth to the one active process.

Typically, SANs have multiple channels (see Figure 4.3). Each channel manages a set of disks in the array. This setup allows SANs to handle multiple requests for data more efficiently. Rather than force all data to move through a single controller, the SAN can move data through several controllers in parallel.

To optimize data flow, administrators split data that needs to be accessed at nearly the same time across different controllers. For example, data warehouses make extensive use of indexes and tables of data. Indexes can quickly pinpoint the locations of specific pieces of data, such as the number of products sold in the Northeast region in the fourth quarter of 2003. That location information is then used to actually find the data on the disk and retrieve it. To keep index operations from contending with data-retrieval operations on the same controller, the indexes and data files are kept on separate controllers.

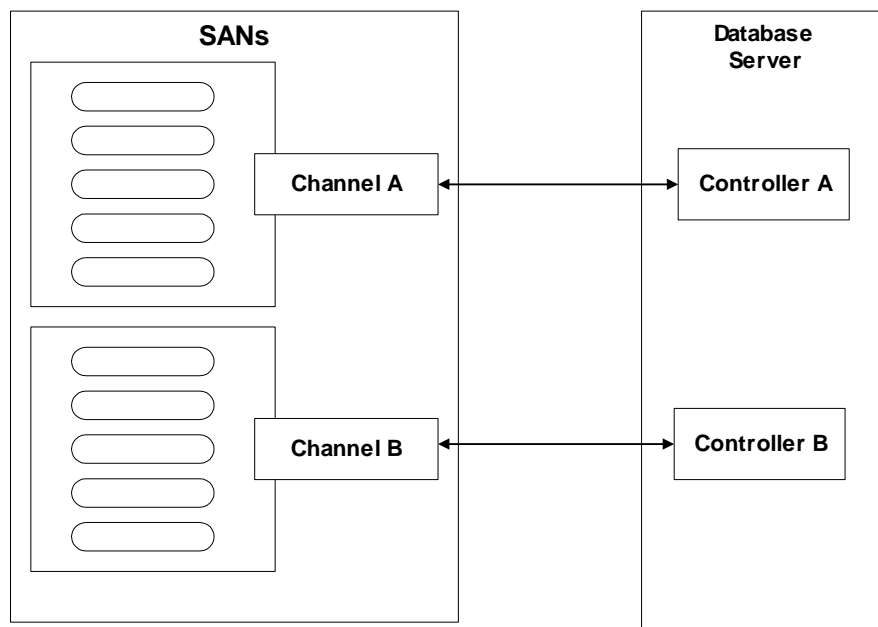



Figure 4.3: Multiple controllers manage data moving to and from disks.

Clearly, parallel operation in the SAN can improve I/O performance just as parallel CPUs on a server improve compute-intensive operations. However, this benefit requires dependencies between infrastructure components such as the SAN and servers.

Details Reveal Dependencies

Digging into the implementation details of network devices often reveals dependencies that are easily missed when working at the higher level of servers and SANS. In the following sections, we'll explore hardware and software dependencies.

 Thanks to Robert Webb of Venturi Partners for providing these examples.

Hardware Dependencies

As Figure 4.3 shows, the SAN channels and the database servers are tightly coupled. Data moves directly from one to the other, which implies that the two must have compatible modes for transferring data, particularly the speed at which data is transferred. The maximum transfer speed of the slower of the two will be the maximum transfer speed of the system. A 250Mbps SAN channel will have an effective throughput of only 40Mbps if the servers have 40Mbps controllers.

High-speed network devices and I/O equipment are limited by the speed of other devices with which they communicate. Changes in IT infrastructure should be assessed at the system level, not just the level of individual components, to identify dependencies that will limit performance of equipment.

Software Dependencies

Sometimes the dependencies between elements of networked systems are less obvious. Consider the relational database management system (RDBMS) for a data warehouse. Many RDBMS vendors offer parallel processing functions to improve performance. Parallel query processing features take advantage of multiple CPUs on a server to divide a user's query into multiple tasks that execute concurrently on different CPUs. Parallel partitions split data in a table into multiple chunks called partitions. From a user's or developer's point of view, all the data is in a single table; the RDBMS is able to spread the data over multiple disks by putting each partition on a separate disk (see Figure 4.4).

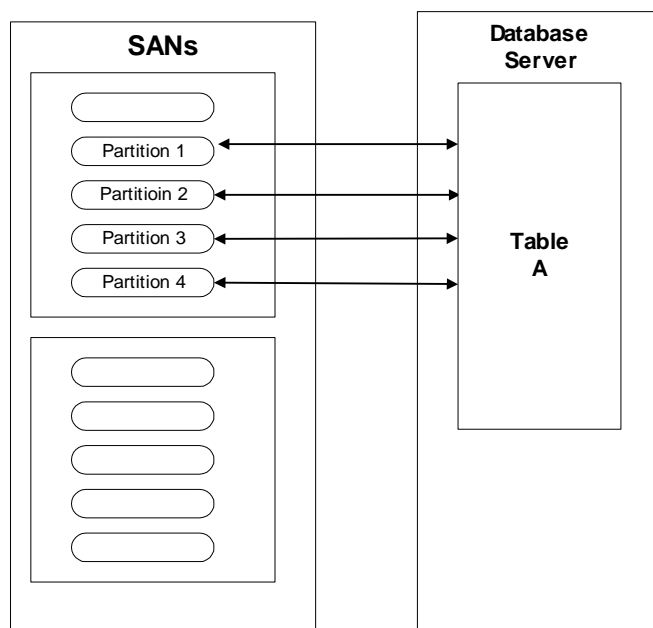


Figure 4.4: Partitioning is a software-based technique that takes advantage of parallel disk operation to reduce the time required to respond to queries and manipulate data.

In general, the RDBMS with parallel processing features will break down a task in such a way as to have one subtask for each CPU. A database server with four CPUs will break a task into four separate processes, and each process will execute on a separate CPU.

Again, problems can occur when the elements of the overall system are not configured properly. If the CPUs are too slow to process data-intensive queries, CPU utilization will reach 100 percent. Because the task has been parallelized to four subtasks, all four CPUs could reach 100 percent utilization for a single query. This behavior effectively blocks other users from executing queries or data load operations.

When hardware is not properly configured, software features designed to improve performance can introduce unexpected performance problems.

Rippling Effects of Misconfigured Systems

Misconfigured systems can result in suboptimal performance, poor user adoption of applications, costly changes to other parts of a system to improve performance, and, ironically, the purchase of additional hardware to achieve expected throughput. (Of course, additional hardware just compounds the problem by adding yet another element that needs to be properly configured.)

For example, consider a high-speed SAN coupled with a server using slow controllers. The decision to purchase that major piece of storage equipment was probably driven, in part, by the need to improve I/O performance on the data warehouse. System designers need to consider the controllers on the database server. Are high-speed controllers available for the server? The controllers need to move data along the internal data bus on the server as well as to and from the SAN. If the internal data bus cannot support high-speed controllers, the server needs upgrading. Some database vendors license RDBMS software based on CPU performance, so upgrading the server could entail simply upgrading a software license. The cost of solving an I/O performance problem can quickly move from the realm of replacing a single storage device to performing major hardware and software upgrades (see Figure 4.5).

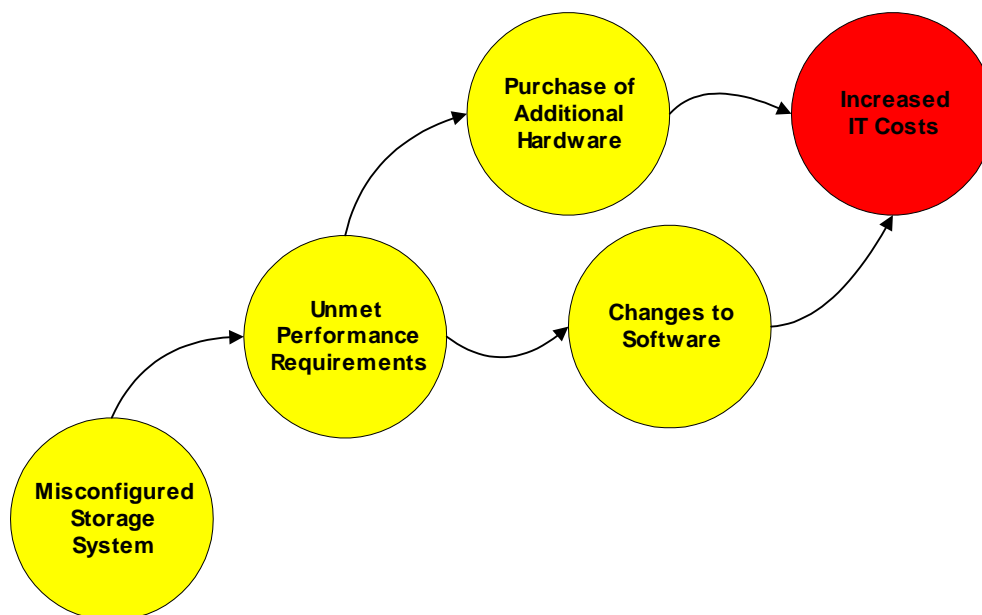



Figure 4.5: *The ripple effect of misconfigured systems often includes unmet requirements and attempts to fix the symptoms without understanding the underlying problem.*

 CPU-based licensing of software can significantly increase the cost of hardware upgrades. Be sure to include these costs when evaluating system upgrades.

In the database partitioning example, the impact of improper configuration can reach end users. As noted, partitioning features of relational databases create processes that can monopolize CPUs and delay the execution of other users' operations. Business intelligence systems are designed for rapid response to ad hoc queries—even large, data-intensive queries—so anything that slows response time undermines system design. Users expect responsive reporting from business intelligence systems, and poor performance will send executives, managers, and analysts to other sources for their information (see Figure 4.6).

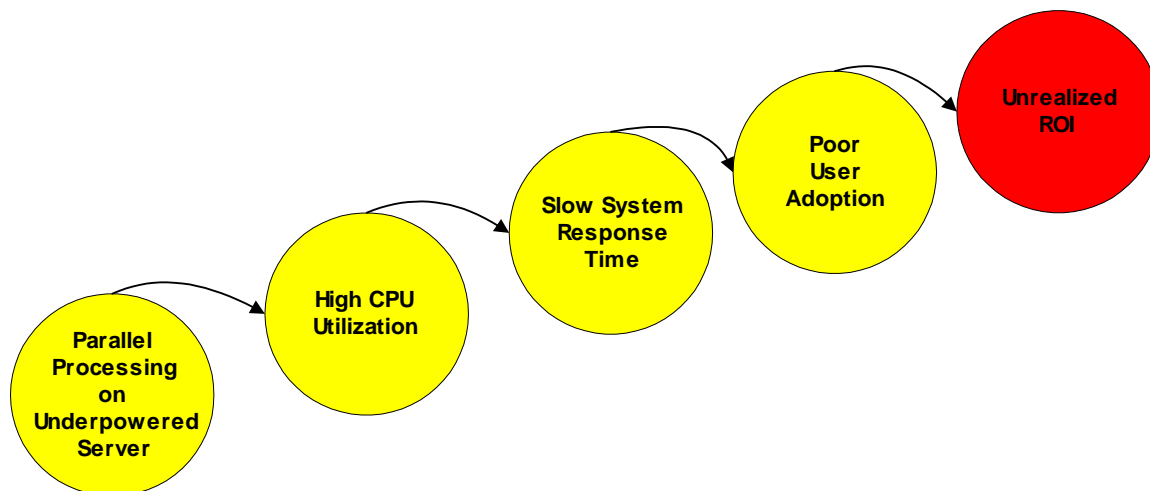



Figure 4.6: Misconfigured systems can undermine user adoption and limit the ROI realized by a system.

Clearly, the ripple effects of change can extend far beyond the server room and affect business operations. Fortunately, ECM techniques provide the means to manage change in this highly distributed and interconnected environment. Before turning to discuss how you apply the ECM model to system configuration management, let's outline the primary goals of system configuration management.


 See Chapter 2 for a detailed discussion of the ECM model.

Defining the Goals of System Configuration Management

System configuration management addresses both business and technical goals. The five main categories include:

- Accounting
- Performance monitoring
- Security
- Fault management
- Configuration management

Together these categories address the main problems involved with keeping an IT infrastructure operating.

 The International Telecommunication Union (ITU) has worked to define the characteristics of these five functions. You can find more information about these topics at <http://www.itu.int/home/> (membership is required to access some information).

Accounting for System Services

Accounting in the area of system configuration does not duplicate the efforts of a finance department; it complements them. As IT departments become responsible for charging back costs to customer departments, it is essential that they track who is using resources and how much those resources cost. Accounting for system services also supports internal IT operations such as:


- Software license compliance
- Upgrade/end of life planning
- Redeployment
- Lease, warranty, and other contract management

It is a costly oversight to simply not perform accounting for infrastructure assets. According to the Gartner Group, organizations that implement asset management systems save as much as 30 percent per asset in the first year and an additional 5 percent to 10 percent in annual costs for the next 5 years per asset (Source: “Enterprise Management ROI and Cost Reduction in 2003”—*Gartner*, November 2002—by M. Nicolle, K. Brittain, and P. Adams).

Charging for Services

Charge backs to customers present a number of challenges. First, much IT infrastructure is a “common good.” Like roads, parks, and police services, the benefits are shared across the community and it is impossible and impractical to measure individual’s use of each service. Costs are allocated according to some measurable use, such as the number of users in a department or the IT budget for other services.

Second, IT departments procure equipment and services from multiple vendors. Sometimes equipment and services are purchased; other times they are leased. Some carry warranties; others do not. Some software licensing fees are based on CPU performance; others are based on named users. Some items carry maintenance agreements; others do not. The wide variety of vendor arrangements requires the implementation of workflow processes and document management practices to manage change.

 Chapter 5 discusses document management and its relation to ECM.

Third, IT services and users change constantly. New applications are brought online. Legacy systems are integrated with new e-commerce applications. Servers that once ran a single application are consolidated into a single server. Segments are added to the network. New services, such as spam filtering for email, are added. The user base changes as well.

Departments cut back on staff and reduce the number of ERP users. Marketing departments implement strategic plans and increase their use of business intelligence systems. To adequately account for the dynamics of IT use, accounting systems need to allocate charges on a short-term basis (for example, monthly). Estimating costs annually during budget development will not accurately reflect the use of IT services.

Software License Compliance

To remain in compliance with software licenses, IT needs to know which software is installed on servers and PCs and, in some cases, how many users are using the software. Automated tools can help inventory PC-based applications. In addition to providing accurate accounts of standard programs, these applications can identify non-standard software that has been installed on PCs without approval. Maintaining compliance on server software can be more difficult.

When software is licensed by CPU, it is important to track any server upgrades or downgrades. Replacing a single CPU server with a 4-processor server can knock you out of compliance. However, redeploying a 4-processor server and replacing it with a 2-processor server would not necessarily put you out of compliance—but could leave you paying more than you should be.

When server software is licensed by named users, administrators should have ready access to changes in a user's status. Enterprise directories and single sign-on servers can provide information about new and terminated employees and contractors. (Single sign-on systems allow administrators to grant or revoke access to multiple applications from one point, so reports on changes by application can simplify ensuring license compliance). When these systems are not in place, administrators typically depend on manual methods to track changes.

Planning Upgrades and Retiring Equipment


Accounting systems provide the raw data needed to plan upgrades and retire equipment. With a database of equipment specifications, IT can project the useful lifetime of an element in the infrastructure and estimate the baseline cost of maintaining the current level of service over a number of years. When combined with utilization information, the accounting database can help with redeployment decisions.

Redeploying Assets

There are several reasons to redeploy assets, including:

- **Eliminating underutilization**—Many application servers are underutilized. During the late 1990s, the conventional wisdom was to install a new server for every application, such as Web servers, email systems, databases, and content-management systems. In some cases, this setup eased software administration by eliminating conflicts between applications running on the same machine and minimizing the number of applications affected by maintenance or a server failure. The downside was that this practice led to a proliferation of servers with excess capacity. It also increased the number of servers that had to be locked down and patched to maintain a secure environment.
- **Reducing licensing fees**—Many organizations are now consolidating servers and running multiple applications on a single server. Doing so can reduce OS licensing costs. Although the total number of users or CPUs on a consolidated server may be greater than on individual servers, the marginal cost of adding those users or CPUs to an already licensed server is less than the total cost of individual licenses for multiple servers.

- Changing OS platforms—In some cases, it is not the hardware but the software that is redeployed. Concerns about security and license costs are leading some organizations to consider deploying applications on the open source platform Linux. Although many Linux distributions are free, enterprise-scale versions of Linux are not. Licensing is only one part of the cost of ownership equation, and, to date, there is no consensus on the relative cost of Linux versus proprietary OSs.

 Many vendors offer their perspective on Linux and its role in the enterprise. For more information, see “Total Cost of Ownership of Linux in the Enterprise” at <http://www-1.ibm.com/linux/RFG-LinuxTCO-vFINAL-Jul2002.pdf> and “A Closer Look at Linux” at <http://www.sun.com/2002-0319/feature/>.

 There is still debate about the total cost of ownership (TCO) of Linux versus other OSs. For a couple of perspectives, see “Linux TCO: Less Than Half the Cost of Windows” at http://www.ciupdate.com/article.php/10493_1477911 and “Linux Servers: No ‘Silver Bullet’ for Total Cost of Ownership” at <http://www.microsoft.com/windows2000/docs/od1035.pdf>.

Managing Leases, Warranties, and Other Contracts

The “do more with less” edict that has come down to many IT departments is driving administrators and managers to save wherever possible. One way to prevent unnecessary expense is by managing leases, warranties, and other contracts effectively. For example, leases may differ on which party is responsible for maintenance and repair of equipment. Having up-to-date access to lease details for all leased equipment is essential to avoid covering the cost of services for which the organization is not responsible. Similarly, exercising warranties can help minimize service and repair costs.

Accounting for enterprise assets is not the core function of IT and, frankly, many administrators would rather have nothing to do with it. Nonetheless, to support change management with respect to IT infrastructure, we need asset accounting. Ideal ECM systems offer functions for allocating costs, supporting upgrading and redeployment decisions, determining end of life and replacement cycles, and managing leases, warranties, and contracts.

Monitoring System Performance

Complex IT systems take much effort to design, install, and configure. It would be nice if after all that work they could run by themselves, but, marketing slogans aside, it just doesn’t happen. These systems need regular monitoring. In particular, administrators need to monitor systems for:

- System load
- Resource usage
- Security breaches

Measuring System Load

A simple phrase like system load can mean many things. When monitoring a server, administrators will check the number of processes executing, the number processes waiting for resources, and the time required to respond to a request, such as to display a Web page or retrieve a result from a database query. When measuring the load on a network, administrators monitor bandwidth utilization (how much data is moving through the network or a segment of the network) and latency (how long it takes to move a packet of data from one point in the network to another). To monitor the load on a custom application, systems administrators require custom key performance indicators. For example, an ERP may be monitored for the number of users, the number of transactions processed, or the time required to execute particular tasks, such as generating a report. Monitoring applications, such as the Windows Performance Monitor that Figure 4.7 shows, capture and display information about a variety of key indicators.

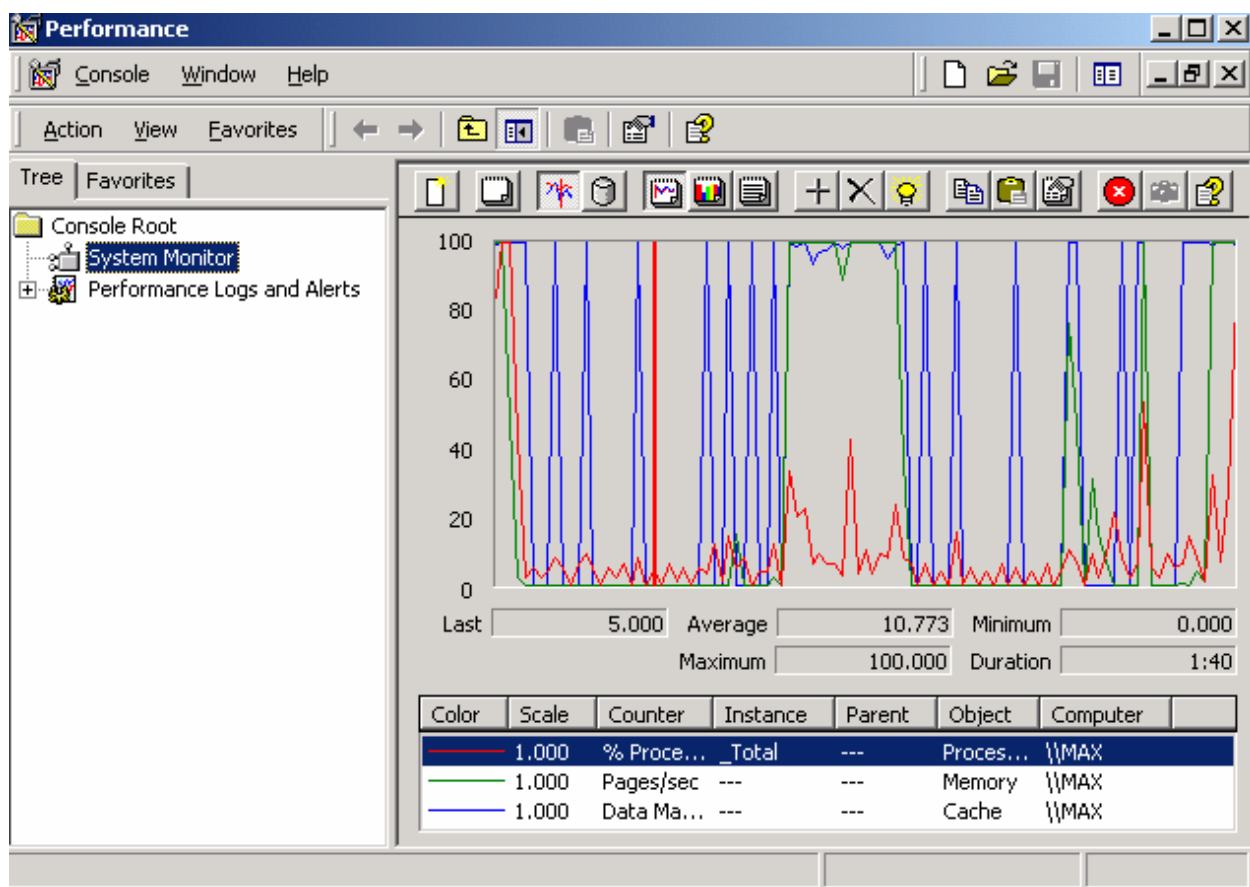


Figure 4.7: OS performance monitoring programs track an array of measures for monitoring processors, memory, I/O, network operations, and sometimes applications.

A common aspect of performance monitoring, regardless of the type of system, is the need for baseline measures, which allow administrators to track trends, identify periods of unusually low or high utilization, and, to some degree, forecast loads. From an enterprise change perspective, administrators need access to baseline information as well as measurements over time that require monitoring processes to be running and logging information. These processes are ideally managed in an ECM system.

Measuring Resource Usage

Administrators can gain a quick picture of overall system load by checking a few performance statistics, such as CPU utilization, free disk space, and total bytes per second transmitted and received on the network. These indicators give an overall assessment of system performance, but may not be sufficient to identify problems.

For example, if CPU utilization is unusually low, the cause could be one of several factors, including fewer users than normal. Not all problems are that easy to identify. The problem may lie with another resource on the system. A slow disk can cause processes to wait while data is retrieved. I/O-bound processes such as these do not use the CPU while waiting. Similarly, a low-bandwidth network can keep processes waiting while data is sent to and retrieved from other servers.

Hardware systems are like production lines. The overall operation is limited by the speed of the slowest step in the process. To identify bottlenecks, administrators need information about the performance of particular resources:

- CPUs
- Disks
- Network interfaces
- Network hubs
- Routers
- Memory
- Caches

Within an ECM framework, administrators will need access to both high-level system load performance statistics and fine-grained resource-level statistics.

Maintaining a Secure Infrastructure

Maintaining a secure infrastructure is one of the most challenging aspects of system management. Accounting and performance monitoring are well-defined operations and are largely in the control of IT staff. Security deals with “what might happen” and the potential list of problems is long. Hackers might:

- View or copy confidential information
- Destroy data
- Use resources for malicious purposes
- Masquerade as a legitimate user
- Install viruses, worms, Trojan Horses, and other destructive programs

Although enterprise security is well beyond the scope of this book, ECM practices are an important part of an overall security practice. Changes in hardware and software can introduce vulnerabilities in a number of ways, three of the most common vulnerabilities are the result of default configurations, software bugs, and unnecessary services:

- **Default configurations**—Software applications sometimes have default configurations that should be changed upon installation. The VAX/VMS OS, for example, used to ship with three privileged accounts set with default passwords. Past versions of Oracle installed with a privileged account and a default password. In both cases, someone with a little administrative experience with either application could gain access to other systems.
- **Software bugs**—Installing or upgrading an application can introduce vulnerabilities to an organization. A common hacking technique is to generate a “buffer overflow” in an application. Doing so basically entails sending more data to an application than it expects and causing data already in the application to be overwritten. Sometimes this attack only destroys the overwritten data. In other cases, hackers replace data with instructions to perform an operation that ultimately leads to more severe damage.

In the spring of 2003, a bug in Microsoft SQL Server allowed hackers to gain sufficient control of servers to have the systems flood the Internet with bogus traffic and effectively shut down Internet services for some organizations.

- **Unnecessary services**—OSs, including Microsoft Windows Server, Linux, and UNIX, install commonly needed services by default. Although this configuration saves administrators time during installation, it creates potential points of entry for hackers. Bugs have been found in basic services. If services are not needed, they should not be installed.

The solutions to these three problems are basic:

- Change default passwords
- Install corrected versions of the program or parts of the program known as patches
- Install only services needed by a server

In all cases, administrators must log the changes and applied patches and services in a change-management system. When a vulnerability such as the SQL Server bug becomes known, organizations with complex infrastructures will need to quickly identify and correct potentially affected systems.

Keeping Systems Updated with Patches

Keeping track of patches can be an administrative nightmare. First, one needs to track all the software in an organization, including everything from email clients on users’ desktops to the ERP system that is the backbone of day-to-day operations. For each piece of software, administrators must know the precise component and version number (not just Oracle 9, but Oracle Net 9.2.0.1.0). Patches vary by OS platform, so this information must also be tracked. Of course, the patches themselves must be tracked too.

In addition, administrators must stay updated on new releases from vendors, or, in the case of open-source software, from development teams. When patches are released, administrators must understand how the patch will affect the application to which it applies as well as the systems that are dependent upon that application. Although vendors do their best to correct problems rather than introduce them through patches, occasionally a fix breaks working systems (see Figure 4.8).

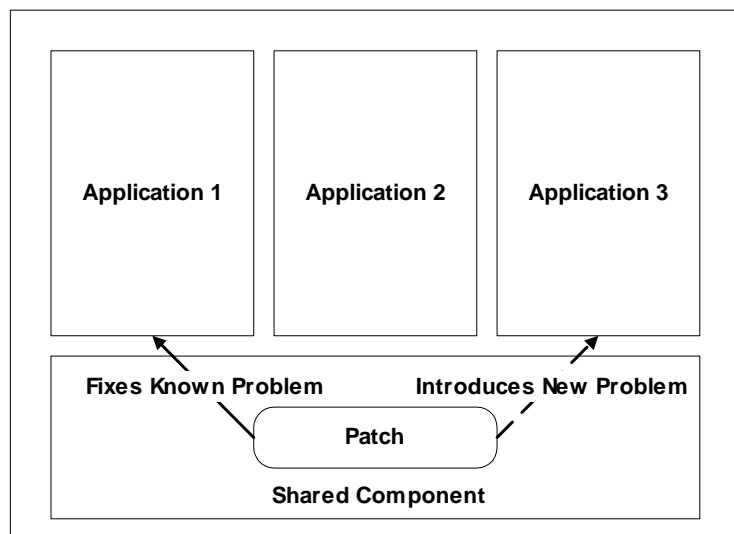


Figure 4.8: Patching a component used by multiple applications risks fixing one problem at the expense of introducing another.

An ECM system allows administrators to track software details as well as the results of applying patches. A simple note in the ECM repository can save administrators hours of installing and uninstalling a patch that does not function correctly in their environment.

Monitoring Suspicious Activity

ECM systems support security monitoring activities. Specialized applications, such as intrusion detection systems and file change-monitoring systems, generate a great deal of information in log files. In most cases, administrators purge log files on a regular schedule, but some should be kept in the change-management system, including:

- Baseline logs when security applications are first executed
- File change detection logs generated after an application is installed on a server
- Intrusion detection logs generated during a security breach

Entries in the ECM system should also include narratives that describe the reason for saving the log and responses to suspicious activity.

Logging Vulnerabilities, Breaches, and Responses

Many organizations are understandably reluctant to discuss security breaches, but that should not stop administrators from sharing details of vulnerabilities, breaches, and responses. ECM systems should be used to record these details and provide a single point of access to that information for internal staff. This record is important in large geographically distributed organizations that employ many systems administrators. When an administrator finds or corrects a problem with mission-critical software, such as server OSs and email servers, the administrator can save others time and duplicated effort by recording and making available this information.

Security depends on knowledge of your infrastructure and its vulnerabilities. ECM systems should track much of the information administrators need to prevent and respond to security breaches.

Fault Management

Fault management deals with identifying and correcting problems in the IT infrastructure. The starting point for correcting problems is to understand exactly how the system *should be* functioning. For example, traffic between any two segments of a three-segment network should be routed directly between the segments. In the case of failure in the network between two segments, the traffic is rerouted through the third segment as Figure 4.9 shows. The problem may not be immediately apparent because of redundant paths through the network. Recognizing there is a problem is the first stage of fault management.

The basic steps of fault management are:

- Identifying a failure in the system
- Isolating the effects of the failure
- Correcting the failure

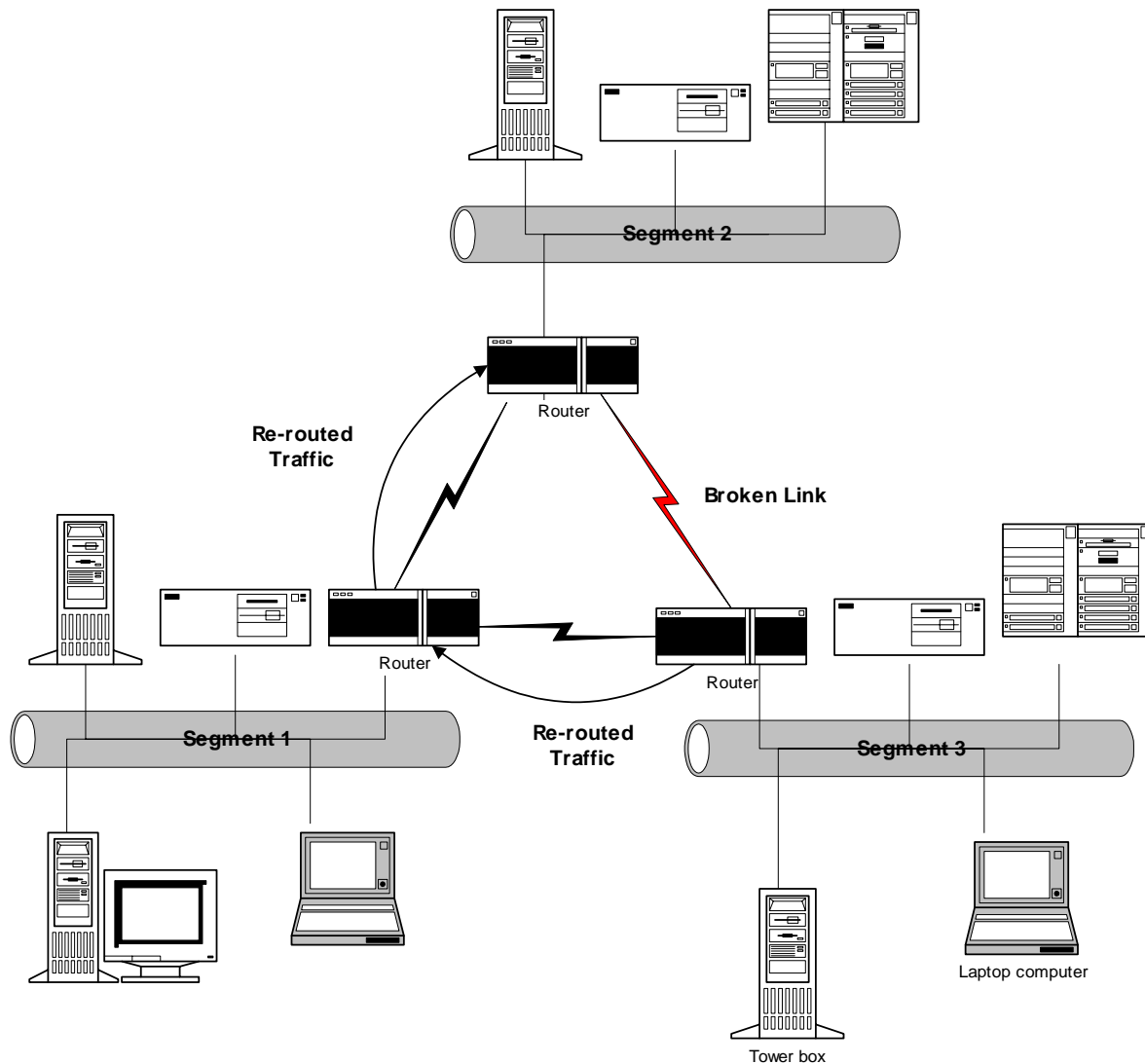


Figure 4.9: Network traffic continues to flow between network segments even during a failure.

Identifying Failed Components

When a major component, such as a server or router, has failed, it is fairly easy to identify. Other failures take more investigation into dependencies between components. Consider an e-commerce system that seems to accept a customer's order but returns a standard browser error message such as *This page cannot be displayed*. The problem could be:

- The Web server is down and cannot display a confirmation page for the customer.
- The database server is down and sales transactions are not recorded. The interface component of the e-commerce system, which is waiting for the database server to return, does not respond to the Web server, which, in turn, generates the error message.
- The database server is not down but is out of temporary work space and cannot process any more transactions.

- A network administrator closed a network port on the application server during a routine security review not realizing that developers had recently moved the e-commerce application to that port.
- A Denial of Service (DoS) attack is underway against your site and the network is flooded with bogus traffic.

There are many other possible causes. Problems with networks and distributed systems often show themselves through symptoms far removed from the root cause. For these situations, systems administrators and network engineers need information about assets that make up systems such as an e-commerce or ERP application, and the dependencies between those assets. With this information in hand, they can devise tests to identify the cause of the problem.

Isolating the Effects of Failures

Once the problem component or system has been identified, it may require isolation. In some cases, such as a failed printer, you do not need to isolate the device, although you might shut down the print queue to keep jobs from building up in the queue.

When an essential component fails, backups must be used. Critical servers are sometimes run in clusters sharing the workload between them. If one server fails, the others automatically pick up the work of the failed server. Networks are often designed with redundant paths (as Figure 4.9 illustrates). Again, this design provides for automatically adapting to the failure.

Other failures require manual intervention. For example, business intelligence servers generally do not require automatic failover protection. If a report server is down for a short time, it won't affect short-term business operations. If the failure is prolonged, a backup server may need to be manually configured and put online. In large organizations, an ECM system could provide information about similarly configured servers, their core functions, typical utilization, and other information that can help to identify a substitute server.

When failures are caused by viruses and worms, information in the ECM system can help to isolate the problem. Take the case of worms—programs that copy themselves from one system to another. One way they can spread is through trust relationships between servers. The idea behind a trust relationship is that an OS on server A will allow an administrator account or process from server B to access the resources of server A as if the account were an administrator account on server A. This relationship eases some systems administration tasks but creates a serious problem when one server in the trust relationship is compromised.

Again, information kept about system configurations in an ECM system could be used to identify servers trusted by a compromised server and list the services running on each server. Administrators could then identify which servers are vulnerable to worms that make use of bugs in particular services.

One of the challenges in isolating failures is that the affected servers may not be physically near each other. In the case of trust relationships, the servers could be anywhere on the network (see Figure 4.10).

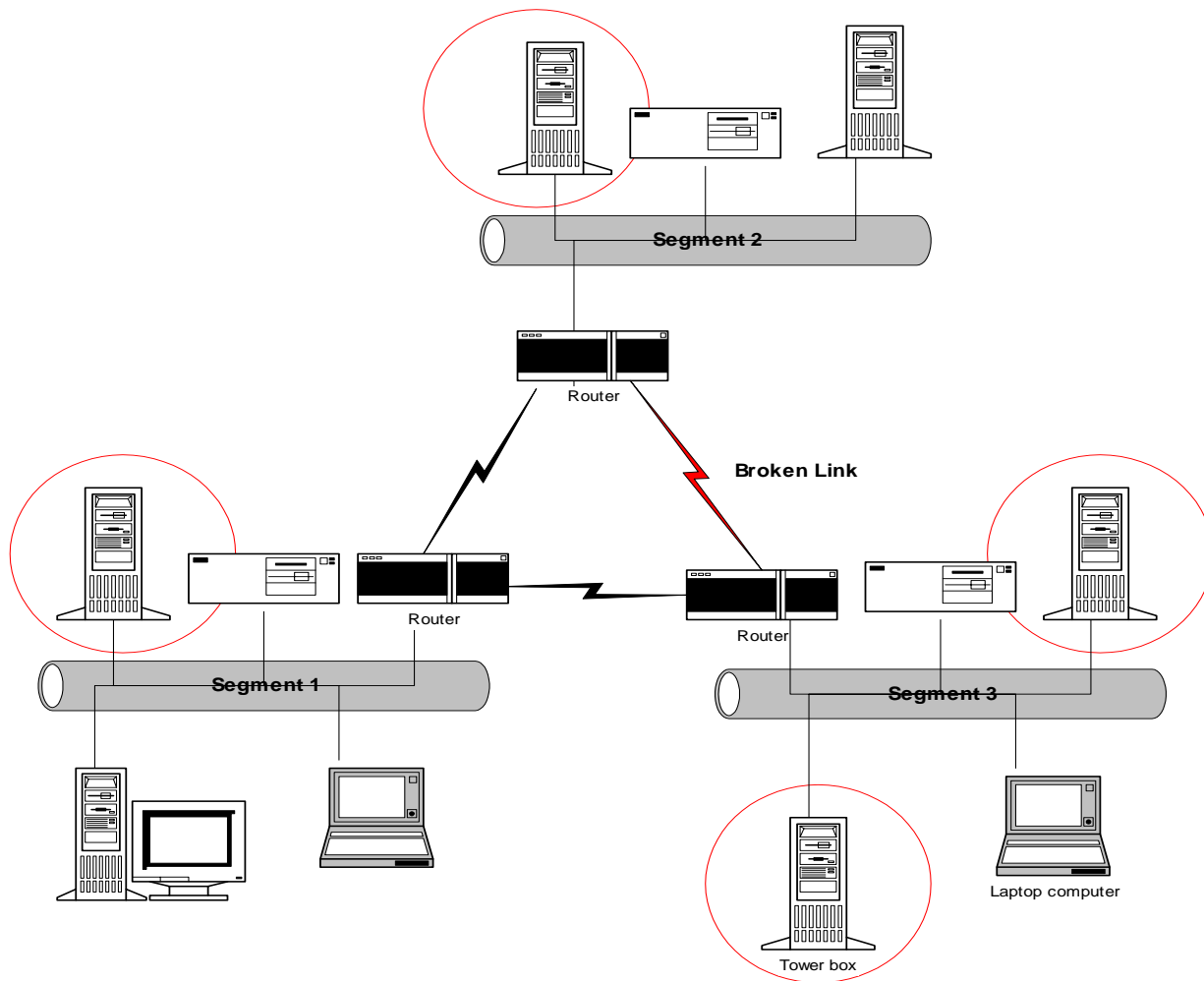


Figure 4.10: Isolating failures in distributed systems requires details about logical relationships, not just physical relationships, between servers. Servers that trust each other are encircled in red.

Correcting Failure

The solutions to failures are as varied as the causes. Information about failures—how they were isolated and corrected—should be logged in an ECM repository for future reference. The repository then becomes a source of basic configuration information as well as a reference on past troubleshooting efforts.

Managing Configurations

The final goal of system configuration management is to keep accurate and precise information about infrastructure assets. Basic characteristics that should be tracked include:


- Device type and logical name
- Physical location
- Version of software elements
- Software patches applied

- Hardware revisions
- History of failures
- Brief description of purpose
- Responsible party and contact information
- Dependencies on other assets

Depending on the type of asset, administrators may also include:

- Performance monitoring statistics
- Security information
- Maintenance history
- Date acquired
- Date placed in current function

The information kept on each device should be driven by administrative tasks—such as maintenance, recovery, and security—and should be standardized across the organization.

 In Chapter 2, we explored the details of the ECM model discussed in the next section.

Modeling System Configuration Management

System configuration management, like other facets of ECM, can be modeled with information about five essential components:

- Assets
- Dependencies
- Policies
- Roles
- Workflows

System configurations are often modeled as collections of elements, networks, and services, so we will discuss that perspective as well.

Defining System Configuration Assets

IT infrastructure assets vary widely in function. Some of the more common network assets are:

- Servers
- Routers
- Bridges
- Switches
- Firewalls
- Wireless access points (WAPs)

These assets support services for the network, such as:

- Dynamic Host Configuration Protocol (DHCP) servers
- Domain Name System (DNS)
- Virtual private networks (VPNs)
- Email
- Instant messaging
- Voice over IP (VoIP)

These assets, while each providing a specific service, have multiple dependencies.

Identifying Dependencies Between Assets

System configuration management should model several types of dependencies. Hardware dependencies entail a physical connection between assets. For example, a server is physically connected to a switch, which is connected to a router, which is connected to a fiber optic backbone, which connects to an uplink to an Internet service provider (ISP), and so on.

Software dependencies occur when applications require other programs to function. Portal servers sometimes require client browsers with Java Virtual Machines (JVMs). Databases are designed for particular OS platforms. Many applications depend on TCP/IP to move information around the network.

Logical dependencies are created by the configurations of systems. The trust relationships supported by some OSs is an example of a logical relationship. A server's membership in a security domain is another example. Clusters of servers are a third example of a logical dependency.

The web of dependencies in even a simple network is too complex to model in precise detail. Your specific needs will determine the information you track. Most of us do not need to know the acceptable voltage range of the power supply in a server. ECM should be driven by pragmatics, not a quest to capture every conceivable bit of information about an infrastructure.

Developing, Publishing, and Enforcing Policies

System configuration management depends on policies to ensure a consistent and manageable environment. Policies and guidelines are developed for:

- Determining user access to applications and network resources
- Configuring firewalls
- Performing security audits
- Selecting OSs
- Configuring Web servers, email servers, and other communication applications
- Carrying out backup and recovery procedures

Roles and Workflows in System Configuration Management

Roles and workflows are closely related. In system configuration management, roles can be roughly broken down into the following categories:

- Element administrators—Responsible for specific devices, such as the routers in the Boston office, the four IBM DB2/UDB databases throughout the company, or the Apache Web server in the local office; element administrators are responsible for the basic operations of these devices and focus on tasks such as configuration, performance monitoring, and maintenance.
- Server managers—Responsible for services such as email and business intelligence reporting; services often depend on configuring and managing a number of different devices, so the scope of a service manager's responsibility spans a wider range than that of element administrators.
- Network engineers—Responsible for the backbone infrastructure on which services depend; network engineers' focus is on the reliable and efficient movement of data.
- Security managers—Unlike the other roles, security managers are not responsible for a device or service. Their focus is on the overall integrity of the system. They work with element administrators, server managers, and network engineers to configure systems, monitor activity, and conduct security audits.

Typical workflows span these roles. Some of the more common tasks are:

- Provisioning user accounts
- Monitoring performance
- Accounting
- Conducting security audits
- Backing up and restoring data

Like software development and document management, system configuration management is an essential part of ECM. Many aspects of system configuration management are captured in the change-management model, but this domain does have some unique challenges.

Understanding the Unique Challenges of System Configuration Management

Managing IT infrastructure poses two challenges not found in software development or document management. First, in many small and mid-sized organizations, network engineers must make changes directly to the production environment without the ability to first try the change in a test environment. Large IT organizations can support a network lab, but even a lab often cannot exactly duplicate the production environment—thus the need for ECM. Network engineers and system designers need to model the affects of change so that they can identify potential problems before implementing a change to the production environment.

Second, system configuration managers must assume that outsiders, and too often insiders, will try to attack their systems. Viruses, worms, and DoS attacks can severely affect day-to-day operations. Again, tracking detailed information about assets and dependencies can help administrators prevent and recover from security breaches.

Summary

System configuration management is evolving. Dependencies between assets are extending further and deeper into organizations, and the business need for reliable network and computing services is increasing. Enterprise-level system configuration management serves multiple needs, including accounting, performance monitoring, security, fault management, and component configuration management. Meeting these needs requires well-defined policies developed and enforced by server managers, network engineers, and security managers. Breaking out of the silo view of change management and addressing the issues presented in this chapter at the enterprise level is the next logical step in the development of system configuration management.

In Chapter 5, we will continue our examination of domain-specific change-management issues with a discussion of enterprise document management considerations. We'll explore how the ECM model enables us to understand the assets, dependencies, policies, and roles involved in document management within the larger picture of ECM.

Chapter 5: Managing Change in Enterprise Content

Organizations are under new pressures to manage enterprise content, including text, image, audio, video, and other unstructured digital assets. Today's enterprises face internal pressures to better manage operations and improve business intelligence, and external pressures from regulatory agencies and new legislation: HIPAA regulations dictate mandatory protections for private health care information; third-party creators of digital assets—such as research documents, online journals, and other references—demand digital rights management; and consumers expect their private information protected, and governments are ensuring that organizations meet those expectations by applying increasing limits on the use of personal data. We are no longer simply dealing with document management—the need to control the lifecycle of digital assets demands adaptive change-management practices and tools (see Figure 5.1).

In this chapter, we'll explore the lifecycle of content and the internal and external organizational factors that influence this cycle. In addition, we'll develop a model of content change management using the framework we discussed in Chapter 2. We will examine how to build policies that support change management tailored to the most important pressures facing an organization. Finally, the chapter concludes with a look into integrating content, system configuration, and software configuration management.

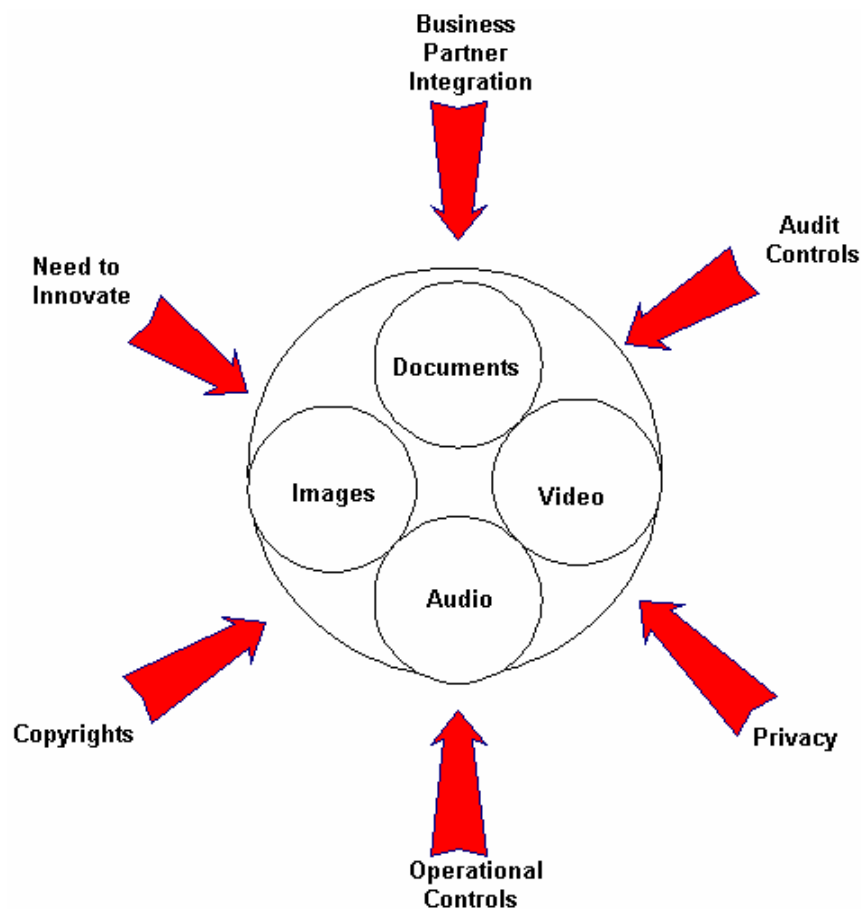


Figure 5.1: Multiple internal and external pressures increase organizations' need for content management and change control.

Understanding the Nature of Content in Today's Enterprise

A new term has emerged in the English language to describe the state of content management in enterprises and across the Web—*infoglut*, which is the condition of being overwhelmed with data and information. Even organizations that practice document and content management are susceptible to infoglut when they step outside their corporate or agency boundaries. Infoglut is the product of the content lifecycle and technologies that surround it. Fortunately, infoglut can be managed. To understand how, we turn to the content lifecycle.

The Origin and Proliferation of Content

Some content is created to serve an end, such as a novel, or as a part of another process, such as a Human Resources memo describing how to conduct performance appraisals. The latter is our primary concern. Organizations create content as a normal part of their operations. The reason to create content is to preserve organizational memory. There are too many details to track, decisions to communicate, and ideas to convey without written discourse.

Although the basic drivers behind content creation have not changed over thousands of years, the methods have. With changing methods, a number of problems have developed, including:

- Growing volumes of content
- Multiple content repositories
- Difficulties in managing unstructured content
- The need to comply with government regulations regarding content

Managing Growing Volumes of Content and Multiple Content Systems

We generate content at a staggering pace. Most organizations have the infrastructure in place to support large numbers of knowledge workers who create and revise content seemingly continuously. Email and instant messaging are supplementing traditional documents and adding to the volume of content that must be managed. The impact of this enabling technology is clear:

- Data on PC hard drives, departmental servers, and enterprise servers is growing at 100 percent per year.
- More than two-thirds of published information is in the form of office documents.
- The deep Web, the online databases that are not accessible to conventional search engines, is 400 to 500 times larger than the surface Web.

There are numerous challenges to managing this much content. Multiple systems are used to store, index, and retrieve content. The Internet is a distributed Web of servers with common protocols for exchanging data. Organizations often use multiple systems for storing content, including:

- Local PC hard drives
- Shared network drives
- Intranet sites
- Document management systems
- Relational databases
- Portals

At one end of this extreme is the local PC hard drive that is generally accessible only to the machine's users. Local drives can be shared across small groups, but ad-hoc sharing of these drives creates more management problems than it solves. At the other end of the spectrum are portals that have the promise of providing a single point of access to multiple systems. For this configuration to be useful, application integration and enterprise search issues must first be addressed.

Managing Unstructured Content

As the volume of content grows, the difficulty of finding content also increases. Structured data is fairly easy to manage using well-developed and proven database design techniques. These techniques allow program designers to create applications that can rapidly find and retrieve specific records and particular pieces of information (Figure 5.2 shows a simple database table).

Relational Database Table					
Primary Key	Code1	Code 2	Last Name	Initial	Phone
ID01	CD1	10	Jones	K	555-1234
ID02	RL1	20	Smith	B	555-3458
ID03	DF9	15	Anderson	R	555-9090
ID04	CD1	39	Williams	L	555-9972
ID05	CD1	19	Johnson	M	555-9811
ID06	RL1	18	Smith	J	555-8081

Figure 5.2: Structured data is organized in fixed structural elements.

For example, a customer inquiry application can quickly retrieve M. Johnson's record if the application has the unique identifier for that record: primary key ID05. Without the primary key, the application can still search for "Johnson" in the last name column and "M" in the Initial column. These techniques do not work with unstructured data.

Content such as images and text is considered unstructured data. It does not neatly fit into relational databases, spreadsheets, or other well-defined structures, and unstructured data does not have clearly delimited attributes such as first name and phone number. Instead, information is spread across text in a free-form manner. Consider, for example, a note in a CRM application:

Received call today from M. Johnson checking on status of order #45876 that should have shipped on Oct. 8. Need to verify status of order and return call (555-9811) by 10/15.


In this example, the name of the customer is not clearly identified. Although we can easily find it, a typical database application cannot. There is no predefined location to put attributes (for example, start name at character position 25 and phone number at position 128). Extracting attributes from free-form text requires specialized applications that can identify the many forms of names, dates, currency amounts, and other entities, as in the following example:

Received call today from <Name>M. Johnson </Name>checking on status of <Order>order #45876 </Order> which should have shipped on <Ship Date October 10, 2003</Date> Need to verify status of order and return call (<PhoneNumber>555-9811</Phone Number>) by <Date> October 10, 2003</Date>.

In this example, XML tags are used to identify features and entities that can then be extracted and loaded into structured databases. These processes are complex and subject to error, and applications cannot always distinguish attributes of the same type (for example, the order date and the ship date).

At best, managing large volumes of unstructured content requires additional processing to extract key attributes. In most cases, management requires developing custom extraction rules and robust data-quality checks to ensure high-quality extraction.

Extracted data serves two purposes. First, it maps elements of unstructured data into a structured representation, making it accessible to existing information-processing applications. Second, it is used as metadata to describe content. The additional metadata enables better content management.

 We'll discuss how you can manage assets using metadata later in this chapter.

The complexity of content management is increasing because of three factors within organizations:

- The growth in the volume of content within and outside of organizations
- The proliferation of content storage and management systems within enterprises
- The need to manage unstructured as well as structured data

External factors are contributing to the growing complexity as well.

Managing Distributed Documents

Documents range from simple one-page memos and short emails to large-scale, multi-segment content with multiple authors distributed over several repositories. Complex and distributed documents present additional challenges:

- Coordinating revisions
- Setting up access control and edit rules
- Repurposing content
- Handling multiple renderings

Complex documents have multiple authors, editors, and users who must approve the content. Their activities must be coordinated to ensure that editors read the most recent version of a document segment, authors maintain an auditable trail of revisions, and those who must approve the content work from final versions.


Access controls typically prevent unauthorized use of documents, but they can also enforce workflow rules. For example, once a document segment has been submitted for review, an editor has to review and revise before the author can make further changes. Similarly, approvers cannot approve a document until a final author and editor version has been released.

XML, which creates semi-structured data, creates numerous opportunities for repurposing content. For example, a standard legal statement can be incorporated into all financial reports created for public review with a single reference. If the boilerplate text is changed, any document that references that text will have the latest version when the document is regenerated. Similarly, a standard product description can be used in sales brochures, Web site content, and other documents through an XML reference. The text of the content remains the same, and the appearance can change as needed.

XML documents are rendered through four methods:

- Cascading Style Sheets (CSS) in Web browsers
- XSL Transformations (XSLT), a transformation language, and FO, a format object specification
- Document Object Model (DOM) and Java (or other programming language)
- Proprietary systems

Although these methods vary in their approaches, they all use basic XML as input and present the content of the XML document in a precisely defined format.

 You can find more information about each method in the XML.com Resource Guide at <http://www.xml.com/resourceguide/>.

When Document Management Is About More than Documents

Document management requires more than a repository with version control to store files. The process now encompasses:

- Multiple repositories logically linked into a single repository through enterprise search and retrieval services
- Support for multiple authors, access controls, and workflows
- Metadata management and, in some cases, metadata extraction
- Semi-structured data for formatting and rendering

A truly enterprise-scale content-management system must address each of these requirements. As we are about to see, external factors are increasing the need for such organization-wide applications.

Growing Demands of Regulation

The second factor affecting the need for content management is the growing trend of regulations governing what content is produced, how it is managed, and who has access to it:


- The Sarbanes-Oxley Act requires a new standard for financial reporting processes
- HIPAA regulations cover protected health care information, including measures health care providers must take to ensure the privacy of patient information
- 21 Code of Federal Regulation (CFR) Part 11 requires auditable processes for electronic submission of regulated content in the pharmaceutical industry
- The Gramm Leach Billey Act (GLBA) dictates how banks can use customer information
- The Securities and Exchange Commission (SEC) 17 CFR Parts 240 and 242 have extended book and records regulations of security brokers and dealers to cover business-related instant messages

The result of these regulations is that organizations are required to maintain more control over content—content that is proliferating at a breakneck pace. The solution is to develop ECM practices that encompass change management and support dependencies across enterprise assets.

Managing External Content and Publishers Rights

Digital rights management adds another dimension to the complexity of managing enterprise content. Content and information licensed from third parties often carry restrictions on use and distribution. For example, access to online databases and journals may be restricted to a set number of concurrent users; users may be limited to a department within a company; the content itself may be accessible only for a set time.

Two commonly used techniques for digital rights management are *containment* and *marking*. Containment uses encryption to restrict access through proprietary programs. Marking inserts metadata tags describing acceptable use of the content. The Extensible Rights Markup Language (XrML) is one markup language used to specify rights and conditions.

 For more information about digital rights management and XrML, see <http://www.xrml.org>.

Although the complexities of content management are growing, its fundamental nature does not change. This is reflected in the content lifecycle.

Managing the Content Lifecycle

Enterprise content follows a complex lifecycle (see Figure 5.3). Some stages are common to all documents, like create and publish; some stages are limited to special cases, such as repurposing; others should be common to all documents but in many cases are not, such as archiving and destroying.

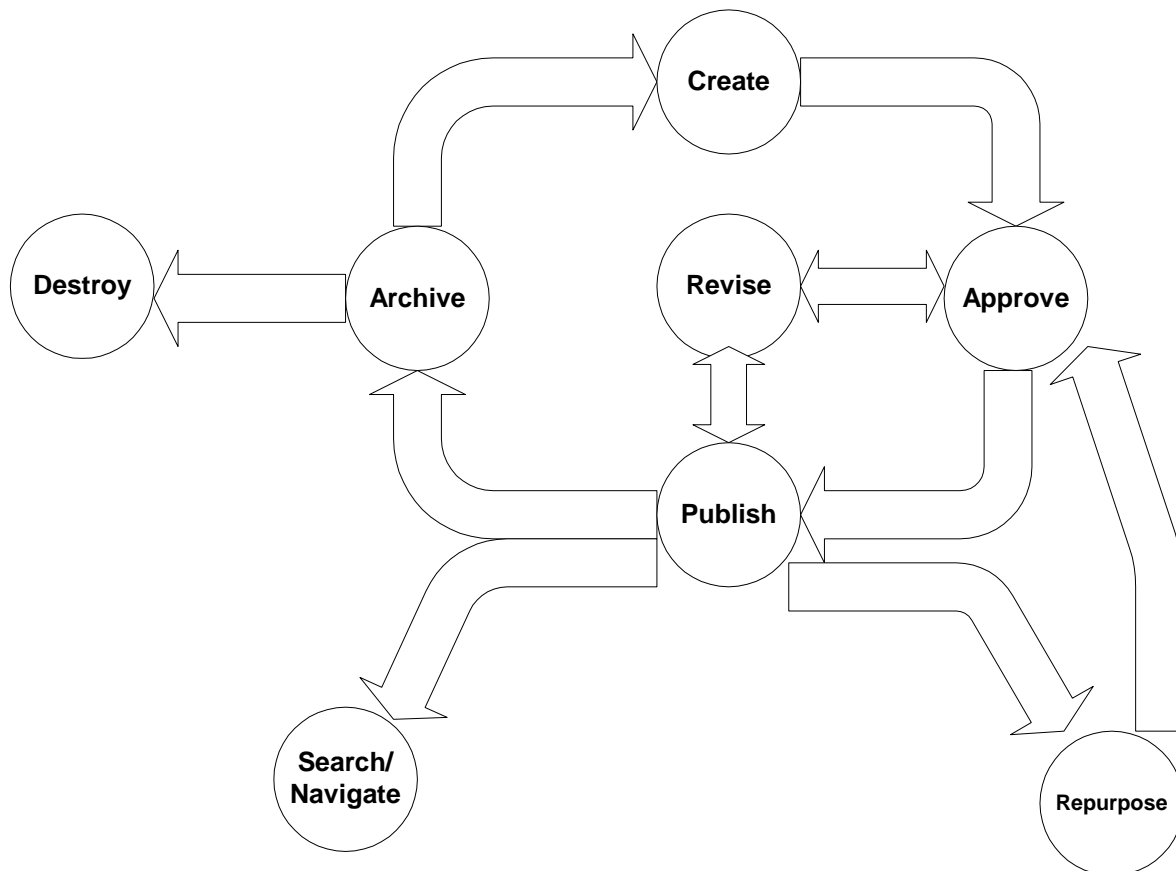


Figure 5.3: The stages of the content lifecycle.

Creating Content

Content originates in a number of ways:

- Produced from scratch, such as a simple email or memo
- Created based on a template, such as a standard project-management document or government-specified report
- Generated as the compilation of multiple pieces of existing content (for example, a product catalog generated by combining structured data from an ERP database with semi-structured product descriptions from an XML database-management system)

Often content is introduced to an organization from outside the enterprise. Email is an obvious example. Documents come from business partners, customers, vendors, regulatory agencies, and other outside organizations. External content that is intentionally acquired—such as business publications, journals, and analyst reports—should be managed like other valuable assets. Unwanted external content, such as spam, requires its own management regime. As we shall see when discussing the model for ECM, policies governing content will vary widely to accommodate these differences.

Approving Content

Fairly small amounts of corporate content require formal approval. Word processing documents, spreadsheets, and presentations are common artifacts circulating in today's organizations. Most of the time, documents are formally reviewed and approved when:

- The content is intended for external use (for example, a sales brochure)
- The content is a part of a legal process, such as contract negotiations
- The content is an official corporate document, such as a regulatory filing
- The content is published as part of a broad or sensitive business activity, such as a corporate reorganization

Implicit in the approval process is a mini-lifecycle of submissions, commentary, and revisions. The final stage of the limited process is approval or rejection. In the former case, the content moves on to the publishing phase of the broader lifecycle.

Publishing Content

Publishing takes on many forms. In simple cases, a file is saved to a shared network drive or added to a document-management system. Descriptive metadata—such as keywords, summary, version indicator, and author information—is added at this point to facilitate search and retrieval. Figure 5.4 shows a typical metadata entry form.

Figure 5.4: Descriptive metadata such as keywords and archive date should be included with published documents.

Publishing may require XML tags to add structure to the document before publishing through a content-management system. It may also require the definition of style sheets or other rendering document to control the format of the final presentation.

Revising Content

Revising content sounds straightforward but is one of the most problematic stages in the content lifecycle. The problems stem from the need to control versions, prevent concurrent editing, and audit changes.

Controlling Versions

Versions are easily controlled within content- or document-management systems. The repository tracks versions, maintains access controls over document versions so that only authorized users are able to create new versions, and provides a single point of reference for the “official” copy of a document. However, the real world does not always fit into this model. Copies of documents are distributed through email, shared in an ad-hoc fashion, and are often edited outside of formal change-control mechanisms. The result is that multiple editions of a document can exist at the same time. Discipline on the part of users is required to ensure that documents revised outside of the defined workflow are not added back to the repository as “official” versions.

Concurrent Editing

Users making concurrent changes are a problem in many applications. In databases, for instance, two users can update a record at the same time and save their changes one after another. Which change is saved depends on who saved the record last. To prevent one user from overwriting the changes of another, database vendors use several options for dealing with concurrent change; some that require complex mechanisms. We do not have that luxury when dealing with documents.

As documents are unstructured, they do not lend themselves to the same types of concurrency controls that work in structured databases. Users are generally left with the option of losing one set of changes, as Figure 5.5 illustrates.

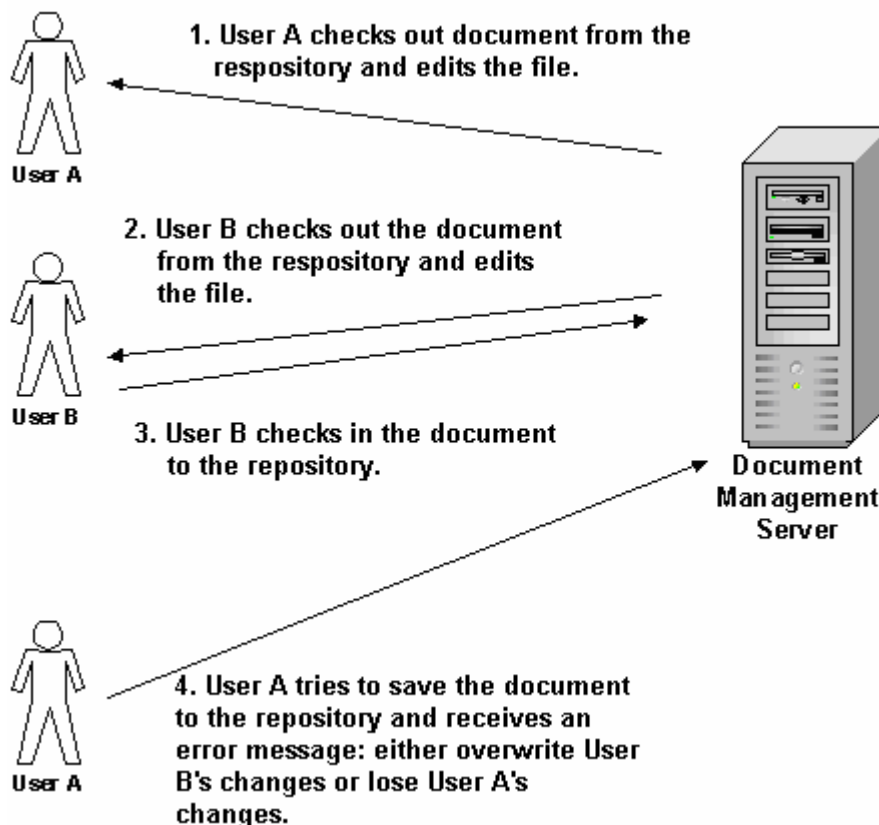


Figure 5.5: Concurrent editing can lead to lost changes.

To avoid this problem, users should:

- Check in documents frequently
- Segment logical documents into multiple small documents
- Use XML to create semi-structured segments that are compiled as needed to generate complete documents



There are technology options to apply annotations to documents when reviewing documents concurrently. These annotations can then be audited and versioned as well.

Auditing Changes

Tracking changes is especially important in regulated industries in which companies must demonstrate the integrity of processes and workflows. Applications maintain change history by either keeping complete copies of each version of a document or by tracking only the differences between the versions. The latter is more efficient because it requires less storage. It does, however, add to the complexity of the document-management system by introducing transactional data that must be managed along with the document itself (see Figure 5.6).

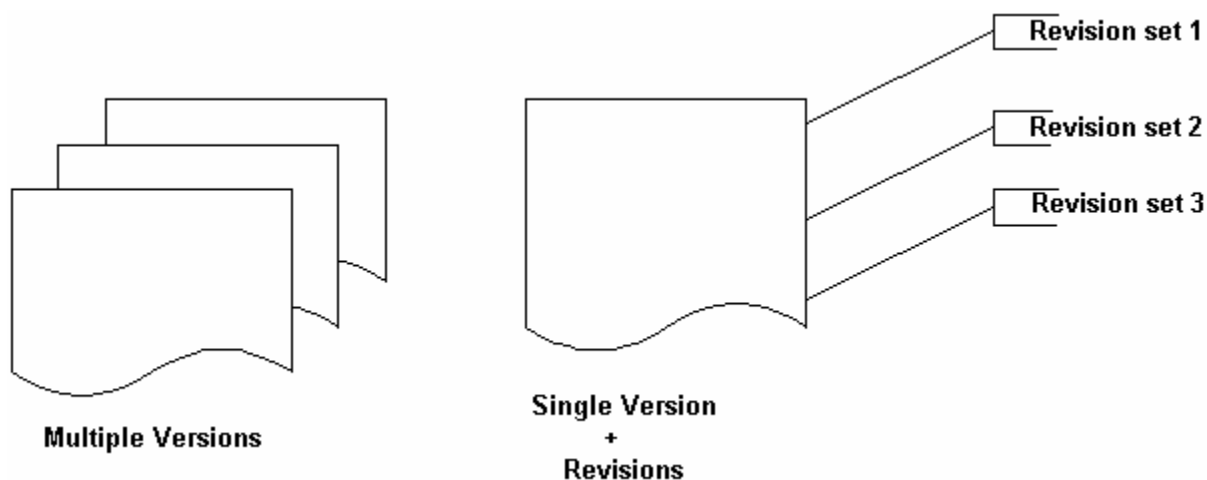


Figure 5.6: Auditing document changes is more efficient when tracking revisions instead of maintaining full copies of documents.

Repurposing Content

Repurposing content is possible when content developers follow a few basic rules:

- Write content in fairly small, logical units
- Use standardized XML tags to identify content structure
- Store content in accessible repositories

Writing Logical Units

In the case of the first rule, “fairly” is the operative word. Obviously, segmenting content into phrases or single sentences is too fine-grained for practical use. Logical units should be based on the semantics of the domain one is writing about.

Using Standardized XML Tags


Repurposing content is difficult without standardized tags. Aggregating content requires finding and retrieving logical units of text (or other unstructured data) much like we do with structured database applications. We need to identify attributes, such as “customer id” or “segment name,” and values, such as “CBA1789” or “short product description.”

Standard tags are defined in schemas, which also describe the structure of content. Listing 5.1 shows a simple example of a product with four attributes: name, id, short description, and long description.

```
<xs:element name="product">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="id" type="xs:string"/>
      <xs:element name="short description" type="xs:string"/>
      <xs:element name="long description" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Listing 5.1: A simple example schema.

This schema defines an element called “product” that has a simple list structure (indicated by the sequence tags). XML query languages allow users to extract target pieces of information, such as the short product description, without retrieving other elements associated with the entity.

 For more information about and a tutorial on XML schemas, see <http://www.w3schools.com/schema/default.asp>.

Storing Content in Accessible Repositories

The final guideline when repurposing and aggregating content is to store content in accessible repositories. These could be document-management systems, XML databases, relational databases with XML interfaces, intranets, or even the Internet. If content is accessible through linking, it is not restricted to a single repository.

Search and Navigate

The longest stage of the content lifecycle is searching and navigating. At this point, content has been developed and refined to the point at which it is suitable for general use. From a content creator’s point of view, the document is done. From a user’s perspective, the work is just beginning. How are users to find documents in repositories of tens of thousands of documents? How are they to know which documents even exist?

Enterprise search systems index content in multiple formats on a variety of source systems. This approach provides benefits similar to Web search engines: users do not need to know about the physical implementation or location of content. This approach also offers similar drawbacks to Web search engines: it can overwhelm the user with hits by including irrelevant content while simultaneously missing relevant content. As Figure 5.7 shows, a single enterprise search system can index content from across the organization.

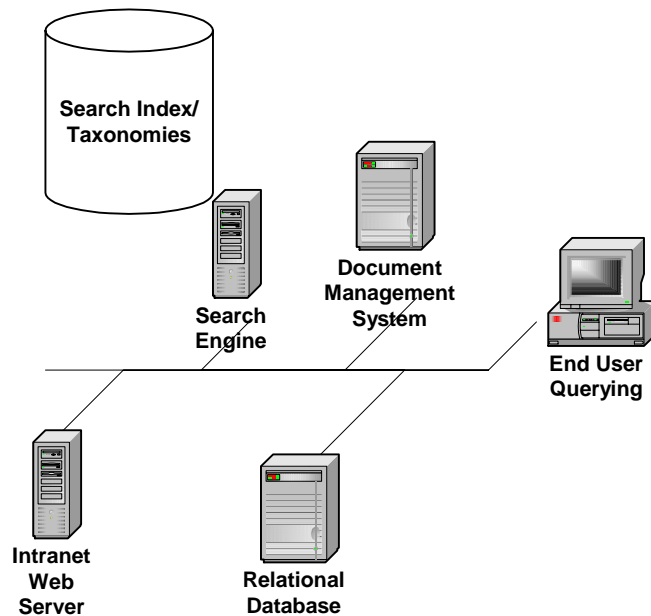


Figure 5.7: Enterprise search applications index content in multiple formats from a variety of source systems.

To improve the quality of keyword searches and to provide an alternative method for finding content, many organizations are now using navigational taxonomies. Taxonomies are hierarchical or linear classifications of content along a dimension. Common dimensions are:

- Geography
- Organizational structure
- Time
- Product categorization

Figure 5.8 shows an example that a retailer might use:

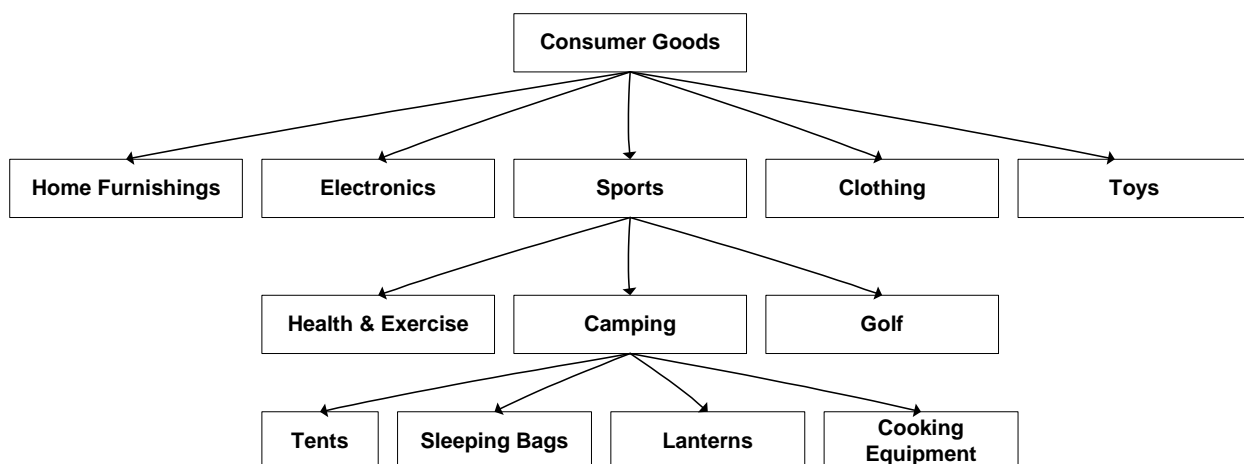


Figure 5.8: A product taxonomy allows users to move from broad, general areas to specific topics.

Combining keyword searching and navigation is a powerful alternative to using either alone. This approach allows users to search within a sub-region of the taxonomy (for example, searching only content in the Camping region and below). How long content remains useful will vary, but eventually its active use will wane and it will move to the final stages of the content lifecycle.

Archive and Destroy

The end of the active life of content is the optional archiving stage. Information that must be retained for legal and historical reasons is archived to long-term storage media (typically, CD-ROMs and tapes). Unlike active content, archived content is not readily accessible through online content repositories.

Much of what we write, edit, and revise has a short useful life and should be eventually destroyed. Of course, many records are covered by statutory retention requirements; other documents may be covered by corporate record retention policies.

 We'll explore record retention policies later in this chapter.

The content lifecycle is complex and so is content change management. Like software change management and system configuration management, the change-management model developed in Chapter 2 provides a framework for managing that complexity.

Elements of Change Management for Content

Modeling change in enterprise content requires:

- Assets
- Dependencies
- Policies
- Roles
- Workflows

Assets

The major types of assets that change in ECM are:

- Documents
- Document metadata
- XML schemes
- Formatting and rendering specifications

There are secondary assets as well, such as single sign-on services, databases, and portals. Their roles extend beyond ECM, and so we will not discuss them in this chapter.

Managing Changes to Documents

Documents are rather generic entities. In the current context, a document can be a:

- Word processing file
- Spreadsheet
- Presentation
- Email
- HTML file
- Dynamically generated portal page
- Collection of aggregated XML-based content
- Image file
- Audio file
- Streaming video

Documents are also dynamic. Authors, reviewers, and approvers can all change documents. A copy that is sent as an email attachment can quickly become outdated. To maintain control over documents and to minimize the impact of obsolete versions, a definitive version of a document should be used.

For example, consider a sales team that is developing a proposal for a client. The original request for proposal (RFP) should be kept as a read-only file in a document-management system, a portal content area of some other shared repository. Team members should receive a link to the RFP rather than an email attachment. This link saves disk space by not duplicating multiple copies of the file, and ensures that any revisions to the document are accessible to everyone on the team. Related documents, such as background material provided by the client, should be saved in the same folder, giving the team a single location for client material.

To effectively control change with documents, follow these general guidelines:

- Use a single content repository for project documents. Primary documents—such as RFPs, draft responses, and background material—should be stored here. Secondary materials, such as news articles about the client, do not require the same level of control. Linking to those documents is sufficient.
- Distribute links to, rather copies of, documents.
- Use version control; include descriptive metadata about each version of a document.
- Define policies dictating who can create, edit, and delete documents within a project area, and enforce these policies with access controls.
- Use distinct folders for different stages of workflow (for example, Draft Folder and Final Revisions Folder).

Other specific guidelines should be developed with regards to dependencies within a content-management environment.

Using Appropriate Document Metadata

Two types of metadata are required for effective document management: content and administrative. Content metadata describes documents with keywords, summaries, author names, creation and revision dates, and categorization information (see Figure 5.9). This metadata is especially useful for search and navigation. Search engines, for example, will rank documents higher if a search term appears in keyword metadata than if it appears only within the body of text. Advanced search forms allow users to specify values for other metadata elements, such as creation dates and author name.

The screenshot shows a dialog box titled "Chapter 5 - Author Draft Properties" with a standard Windows window title bar (help, close). The dialog has five tabs: "General", "Summary", "Statistics", "Contents", and "Custom". The "General" tab is active. It contains the following fields and values:

- Title: definitive Guide to Enterprise Change Management
- Subject: Chapter 5
- Author: Dan Sullivan
- Manager: Laurie Nocella
- Company: Realtimepublishers.com, Inc.
- Category: change management
- Keywords: document and content management; metadata
- Comments: Author draft
- Hyperlink base: (empty)
- Template: RTP eBook

At the bottom, there is an unchecked checkbox labeled "Save preview picture" and two buttons: "OK" and "Cancel".

Figure 5.9: Metadata added to the properties sheet of a Microsoft Word document can be used by enterprise-scale search engines to improve the quality of search results.

Administrative metadata is used for maintenance and compliance operations. Archive or expiration dates identify the end of the useful life of a document. Rights metadata describes copyrights and limits on the use of content. Last change dates are used to implement incremental backups. Administrative metadata will vary according to specific operational requirements.

Tracking XML Schemas

As I mentioned earlier, XML schemas describe the framework of semi-structured documents. The first entry of an XML document typically indicates the XML schema used. For example, the product information that Listing 5.2 shows is based on a schema called catalog-schema.xml.

```
<?xml version='1.0'?>
  <consumer-good xmlns="x-schema:catalog-schema.xml">
    <item category="camping">
      <product-name>Acme Outdoors 6-person Tent</product-name>
      <manufacturer>
        <manf-name>Acme Outdoors</manf-name>
        <manf-id>ACME0133</manf-id>
      </manufacturer>
      <price>198.99</price>
      <short-description>This 6-person tent is a versatile ...</short-
description>
      <long-description>Designed for the camping family, this tent
offers large
          doors, 3 windows, steel stakes and a polyester mesh
ceiling. ...
      </long-description>
    </item>
    ...
  </consumer-goods>
```

Listing 5.2: An example schema.

Schemas are documents and are therefore subject to the same change-management issues as other content. Their close relationship to XML documents introduces dependencies that I'll discuss shortly.

Formatting and Rendering Specifications

Formatting and rendering specifications are like XML schemas in that they are documents that function closely with XML and other semi-structured documents. In addition to managing them as documents, we must manage their referring documents as well.

Content assets are no longer restricted to traditional, self-contained documents. Structuring frameworks, such as XML, and rendering schemas, such as XSLT and CSS, are now important elements in ECM. These new elements greatly enhance our ability to reuse content and manipulate text with greater precision. The cost of these benefits is more dependencies between assets.

Identifying Dependencies in Enterprise Content

Dependencies in ECM emerge from several places, including infrastructures that support content management, the relationship of content to other enterprise assets, and the use of externally controlled assets, such as XML schemas and protocols.

Infrastructure Dependencies

Infrastructure dependencies can be subtle. For example, how should documents be archived so that they can be opened in 5, 10, or 20 years? File formats change and commonly used programs may not be available in a decade. Storage media also changes. Although magnetic tapes continue their long life as a staple of IT infrastructure, media formats change along with their hardware. Any layer of IT infrastructures (OS, file formats, storage media, and encoding schemes) that can change will have an impact on how archived material is used.

Related Asset Dependencies

Content is sometimes closely linked to other content, as in the case of XML schemas and formatting specifications. Documents that use those schemas or specifications are limited to using tags and functions defined in the corresponding document. The more complex schemas and formatting specifications become, the more difficult it is to change them while maintaining backward compatibility. Changing a schema without backward compatibility can force authors to update documents, which, in turn, can initiate workflow and approval processes, which can introduce more changes. These changes ripple through to other dependent assets (see Figure 5.10).

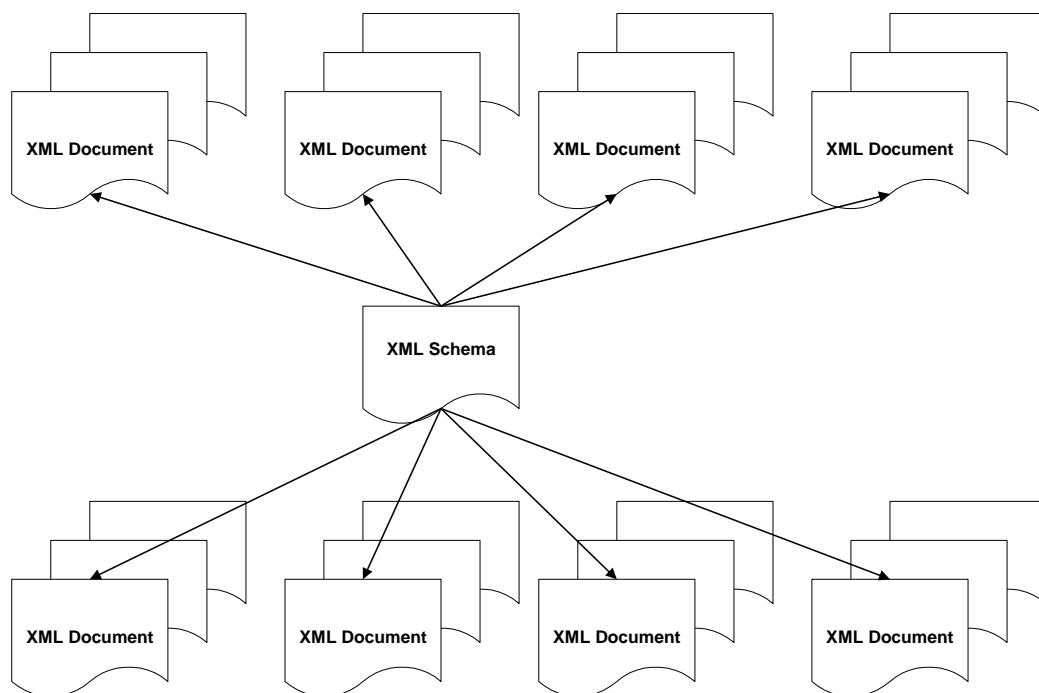


Figure 5.10: Changes to a single XML schema can force changes within many dependent documents.

External Dependencies

External dependencies emerge from two sources: business partners and publicly available standards. Customers, suppliers, and business partners need to collaborate through documents. In most organizations, communication is ad hoc and done through email. Through more efficient methods such as shared document repositories and semi-structured documents, the dependencies become more pronounced.

The trend toward XML-based business communications is clear. There are more than 60 technical committees within the Organization for the Advancement of Structured Information Standards (OASIS), a technical consortium dedicated to developing structured information standards that cover a range of business areas:

- Auto repair
- Biometrics
- Business transactions
- Conformance
- Customer information
- Digital signatures
- E-Government
- Emergency management
- Law (court filings, contracts, e-notary)
- Product lifecycle
- Provisioning
- Tax

These standards will be adopted as they evolve because of the efficiencies they bring. Organizations need to take care in selecting standards and versions of standards to ensure that the standards they choose are used by the business contacts with whom they communicate. These communication partners need to stay in synch as the standards evolve and new versions are released.

Content dependencies cross organizational and technical boundaries. Identifying and modeling them is the first step to successfully managing change in this domain.

Developing Content-Management Policies

Policies for ECM address several areas:


- Roles, audit controls, and record retention
- Tagging rules
- Access controls and digital rights management

Staying in Compliance with Policies

Policies in content management are similar to policies in software development. They identify who (or which roles) can create, modify, and delete content; define templates and schemas; and define new shared folders and other management functions.

Audit controls and record retention ensure compliance with legal requirements and management practices. At a minimum, effective change management requires a system to track which user has made a change and at what time the change occurs. More effective is including a description of the changes as metadata associated with the new version. This metadata can be supplemented by a list of the actual changes derived from a utility for identifying differences between two files. These are progressively more complex controls and only the most important processes will warrant the most comprehensive procedures.

Record retention is also a form of compliance. The object of record retention policies is to ensure that legal and corporate records are kept as long as necessary without creating a save-everything-because-you-never-know-when-you-will-need-it culture. Well-defined policies, such as a schedule of how long different types of documents should be kept, are key elements of a well-governed ECM system.

 The failure to retain records has legal ramifications. Although it is not as popular a topic as other forms of compliance, it should be treated as a legal matter.

Maintaining Consistent Document Structures and Metadata

To ensure that document schemas and metadata are used consistently, develop policies specifying key features. Policies governing schemas should address which version of XML standards are used for a particular application, when and what controlled vocabularies are used for attributes, and procedures for changing schemas. Similarly, metadata policies should identify which attributes are required and which are optional; when controlled vocabularies are used for attributes; and what dimensions or facets are used for creating taxonomies.

Controlling Access and Protecting Digital Rights

Access controls are common in content- and document-management systems. They limit which operations can be performed on an object, such as creating a file in a folder and reading a document, and who can perform such actions. Less common but of increasing importance is the ability to manage digital rights. Digital rights specify how content can be used:

- How many users can access content concurrently
- How long content is available online
- How content can be redistributed and repurposed
- Which users or organization members can access the content

Specific rights will vary with content and providers but organizations should maintain policies to support the protection of digital rights. Policies should be defined to specify functional requirements in portals and other access systems that preserve digital rights.

 For a thorough overview of digital rights management and support architectures, see <http://www.dlib.org/dlib/june01/iannella/06iannella.html>.

Specifying Roles and Workflows

Roles and workflows are the final elements in content and document change-management models. Roles typically reflect common tasks such as:

- Creation
- Review
- Approval
- Administration

These roles, in turn, are used to specify who is responsible for steps in workflows. In addition to the lifecycle described in this chapter, workflows are based upon audit and review processes and administrative activities, such as archiving and backup. As with software development and system configuration, change management in the content- and document-management arenas is best understood and managed along the lines of the ECM model.

Summary

ECM takes a broad view of change within an organization. As we have seen in this and the previous two chapters, domains once limited to silo-based change-management approaches can actually be managed with a single model. Content, systems configuration, and software development share common content-management tasks and requirements. Managing assets has more to do with understanding assets, their dependencies, and their workflows than the low-level implementation details.

The next, and final, chapter explores ECM practices, tools, and benefits. We'll conclude with a look at the future of ECM—what will change and, just as important, what will stay the same.

Chapter 6: Practicing ECM

ECM is more than the sum of silo-based change-management practices. ECM crosses organizational and functional boundaries to integrate assets, coordinate workflows, establish policies, and provide the mechanism to identify and control dependencies. Software development, systems infrastructure, and content management are essential elements of organizations' day-to-day operations. Prior to ECM, these assets were treated as distinct and separate products with their own life cycles and management regimens. Confluences in workflows were more coincidental than managed. Policies were narrowly focused with little consideration for dependencies across functional domains. ECM has changed the framework for change management.

Within ECM, the focus is not on distinct types of assets—such as software, network devices, and documents—but on their common attributes. Chapter 2 developed a model, or framework, for understanding change management at the enterprise level. The model includes:

- Assets
- Dependencies
- Policies
- Roles
- Workflows

Each of these elements is relevant to silo-based change management but is even more useful when combined to span an enterprise-wide ECM framework.

Varieties of Asset Change

Assets are objects that change over time. These include enterprise applications, such as CRM systems and data warehouses; systems infrastructure, such as servers and firewalls; and content ranging from project plans to design documents.

Simple Change-Management Patterns

Assets often have multiple versions. The evolution of those versions varies with the complexity of the asset and the number of participants in the process. Assets can change in a linear fashion; for example, as Figure 6.1 illustrates, an article is written by an author, revised by an editor, and final revisions are made by the author in a linear process.



Figure 6.1: Linear changes are the simplest to manage.

Software development for small, single-developer applications also have a simple change-management cycle. Developers often work in a code/compile/test cycle as depicted in Figure 6.2.

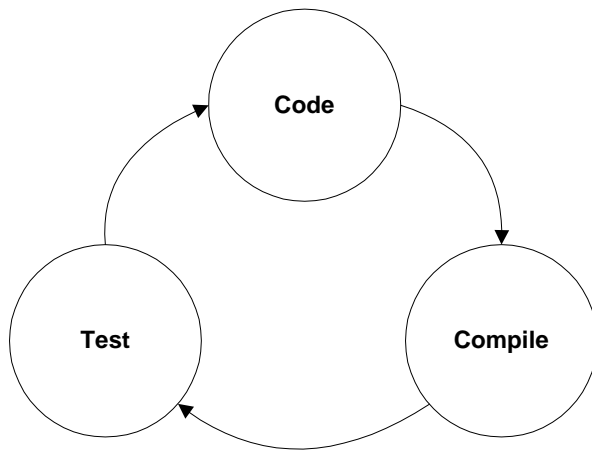


Figure 6.2: Cyclical change patterns occur in small software development efforts.

Complex Change-Management Patterns

Changes to most enterprise assets are more complex than these examples show. Consider a new policy on email privacy from a Human Resources (HR) department. The need for a policy is initiated by some event in the organization, such as the disclosure of a private communication, or as the result of a regulatory requirement, such as the HIPAA legislation governing healthcare records. Staff members from across the organization will contribute to the policy. HR, Legal, and IT provide initial input including objectives, legal requirements, technical constraints, and cost estimates. Figure 6.3 shows the development cycle for this type of process.

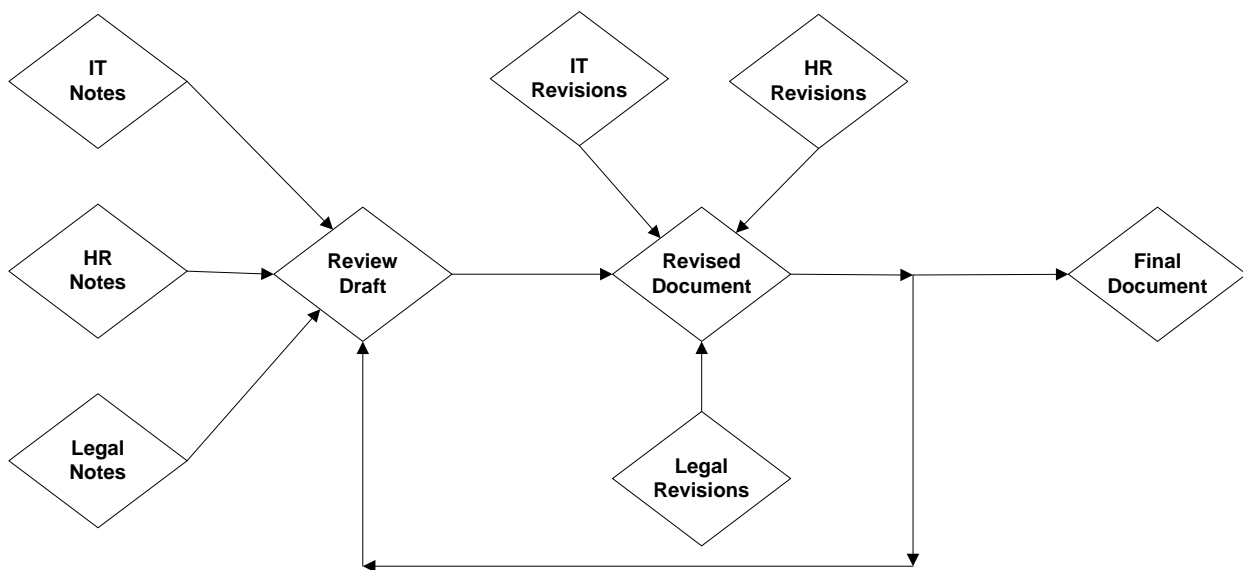


Figure 6.3: When multiple participants and multiple iterations are involved in asset revision, the workflow becomes more complex.

In this example, multiple participants are simultaneously revising a single asset. If each participant revises a version of the document, the changes need to be manually consolidated. Some document-management systems support tracked changes from multiple participants against a single document. In addition to eliminating the consolidation step, these tools provide the basis for effectively auditing changes.

Software development and systems configuration management have similar patterns of complex change. In the case of software, versions vary by function and platform. New versions of applications typically add new features, correct bugs, and improve performance. These changes occur throughout the application. Platform-specific versions typically concentrate changes on low-level services closely tied to an OS, such as file services, process management, and access controls. Coordinating multiple changes across two dimensions—function and platform—further increases the complexity of the management process.

One of the most significant challenges in systems configuration management is the difficulty in duplicating production environments in a development and test environment. Many organizations create network labs to test and configure products, but the cost of implementing and maintaining identical production and test systems is prohibitive. This situation places a premium on understanding the dependencies between systems and their configurations. Configurations often span multiple platforms, organizational units, and functional boundaries. Effective ECM requires an understanding of these configurations in order to foresee the impact of changes to other assets and processes.

Many tools are available to manage each of these change life cycles individually. The limits of these silo-based management tools are motivating the development of cross-domain ECM applications. As with many complex, organizational problems, there are multiple approaches to a solution; however, they are not all equally effective.

Selecting ECM Tools

Selecting an ECM tool entails several steps:

- Assessing an organization's functional requirements
- Evaluating technical features of ECM tools
- Estimating the ROI

Assessing the Functional Requirements of an Organization

Functional requirements span a range of needs. This section describes those requirements that should be considered for all ECM applications.

Determining Asset Types

Some assets are obvious—such as source code for applications, project documentation, and network hardware; others stretch our notion of assets—such as strategic plans, intellectual property, and best practices. Some assets are managed directly in an ECM system—documents and software, for example. In other cases, we manage information about assets, such as the type and configuration of a network router. The first step in determining functional requirements is identifying all the asset types that need to be managed. For each asset type, understand its life cycle:

- How is the asset acquired or created?
- How does it function with other assets of the same and different types?
- How does the asset change over time?
- What priority is placed on effectively managing the life cycle of this asset?

Next, assess the dependencies that exist between assets and configurations.

Assessing Dependencies

Assets depend upon other assets and stages of workflows. ECM tools should support common dependencies, such as software releases that depend upon successful builds, which, in turn, depend upon versions and branches of source code development. Other dependencies will vary by the specific requirements of a project.

Consider a strategic marketing plan. It depends upon product release schedules, which, in turn, depend on the outcome of production schedules, quality assessments, and preliminary marketing research. These schedules and this research are stored as documents that are produced, in some cases, by staff and consultants, and in other cases, by an application, such as an ERP tool. By clearly identifying these dependencies, managers can pinpoint bottlenecks and develop risk mitigation strategies to adopt a strategic plan for unexpected events.

An ECM tool should effectively model a variety of dependencies. In particular, it should:

- Report on assets and workflows affected by a change to an asset
- Allow for multiple dependencies between assets
- Include information about timing dependencies
- Rank priorities of dependencies

With an understanding of both common dependencies and dependencies specific to an organization, the next step is to understand workflow processes.

Understanding Workflows Within an Enterprise

Assessing workflows within an organization is similar to assessing dependencies. There will likely be common workflows, such as in software development, testing, and releasing, as well as organization-specific workflows. When considering idiosyncratic processes, do not limit the discussions to the traditional change-management domains of software, systems, and documents. Areas such as strategic planning, budgeting, product development, and other high-level business processes lend themselves to the benefits of change management.

The discipline of ECM is appropriate to a broad range of operations. Well-designed ECM tools will lend themselves to processes outside the traditional areas of change management.

Evaluating Access Control and Security Requirements

All enterprise applications must address security. Some ECM requirements are similar to those found in other applications. ECM tools should integrate with single sign-on (SSO) services and utilize directory services such as Microsoft AD or other LDAP services. The tool should also support ECM-specific requirements such as controlling users' ability to:

- Create, delete, and modify assets
- Define and edit dependencies between assets
- Create and modify workflows
- Define triggers, such as notifications that an event has occurred
- Change security and access control settings

ECM tools should also enable administrators to assign users to groups, which, in turn, are assigned privileges. In addition to their use in controlling access, roles are also used to define which users can execute steps in workflows, as Figure 6.4 shows.

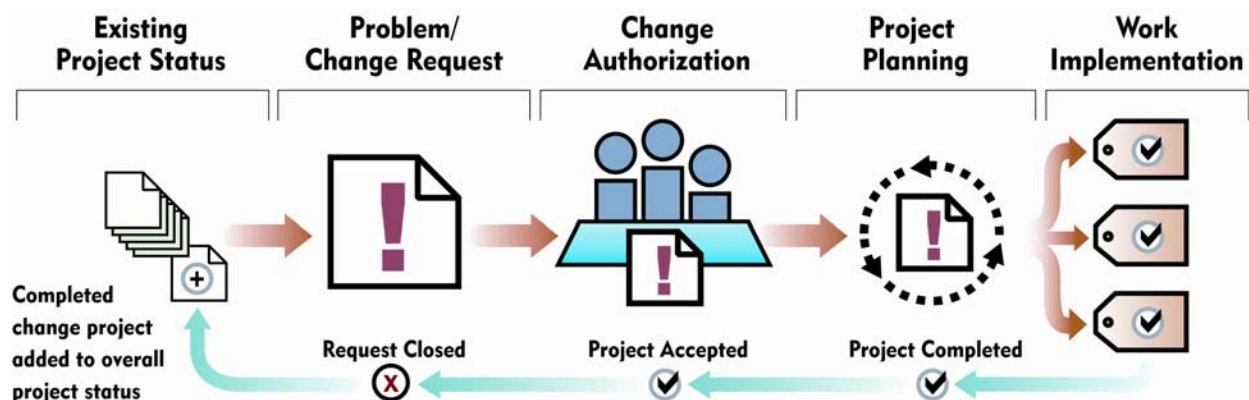


Figure 6.4: Roles serve multiple purposes in ECM applications, including controlling the execution of workflows.

The combination of assets, dependencies, workflows, and their access controls are the elements of ECM, but functional requirements should consider two additional aspects as well: information delivery and legacy application integration.

Delivering Information

ECM systems maintain a wide range of in-depth information. Assets come from across the organization, dependencies span traditional boundaries, and workflows can entail large numbers of users. Information about these assets, commonly called *metadata*, is also found across the organization and in varying forms. Add to this the multiple versions and branches associated with assets, and you can see how an ECM repository can quickly grow with detailed information. This information growth is a doubled-edge sword. On one hand, having ECM details in a single source provides one point of access for information. It also provides the means to integrate information across domains, projects, and operations.

On the other hand, the ECM application can suffer a similar problem to the World Wide Web—with so much information available, it's difficult to find exactly what you need. The key to successful information delivery is identifying specific items rapidly and keeping related items easily accessible. This goal is accomplished in several of ways, including:

- Search
- Navigation
- Visualization
- Structured reports

Search should include basic lookup functionality based on named entities, such as “Finance Department Data Mart—Invoice Extraction Script” and “Boston WAN Segment—Router 3.” It should also include metadata searching—searching by metadata attributes, for example “all routers in the Boston WAN Segment” and “all data extraction scripts modified in the last 30 days.” This type of targeted information retrieval is best suited when you need information about a particular asset, process, or dependency. However, a broader view is sometimes needed.

Navigation allows users to move from general categories to specific instances and vice versa. For example, navigation enables users to start at a high level, such as project documentation, and move down a hierarchical scheme to IT projects, through business intelligence projects, down to a Finance department data mart documentation (see Figure 6.5).

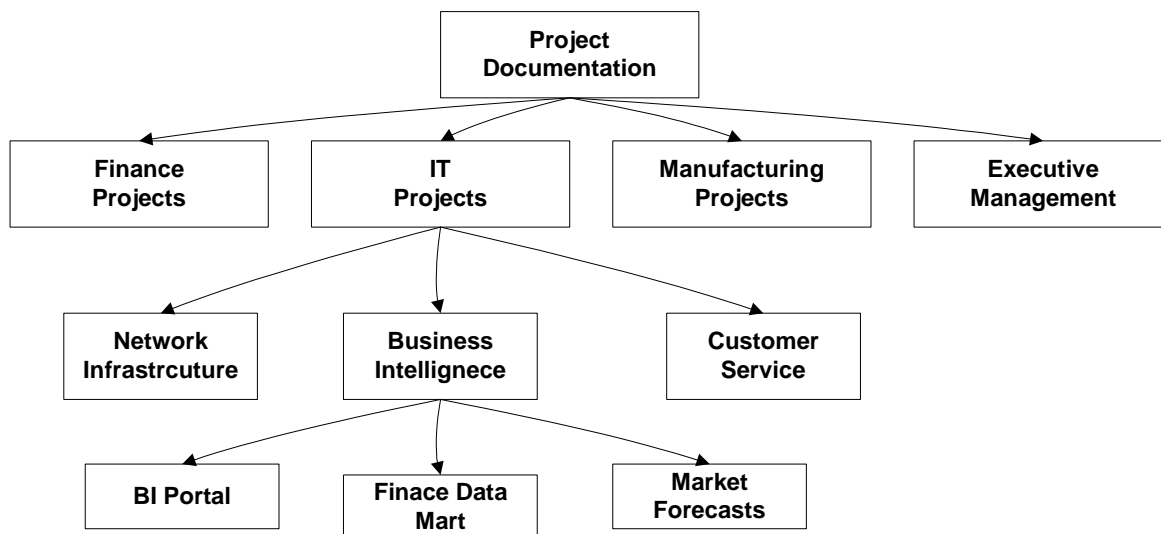


Figure 6.5: Hierarchical navigation allows for information retrieval based on topics rather than search terms.

We know from enterprise content management that no single hierarchy, or taxonomy, will meet all navigation needs. Generally, multiple taxonomies are required. In ECM, hierarchies could be designed around projects, assets, departments, or time periods. Specific taxonomies do not need to be designed prior to selecting an ECM tool. The goal should be to use a tool that supports multiple modes of access to repository data.

Search and navigation are based on terminology. Navigation provides more contexts than search, and multiple taxonomies provide greater depth to context than a single taxonomy can. In some cases, even this information delivery method is insufficient to quickly grasp the context of an asset or workflow. In those cases, visualization tools are called for.

Visualization is a powerful tool for understanding the overall structure and breadth of a topic in an ECM repository. Ideally, a visualization tool provides a picture of the relationship between elements of an ECM repository, such as select assets and the dependencies between those assets. Visualization is also essential for quickly understanding the nature of a workflow (see Figure 6.6).

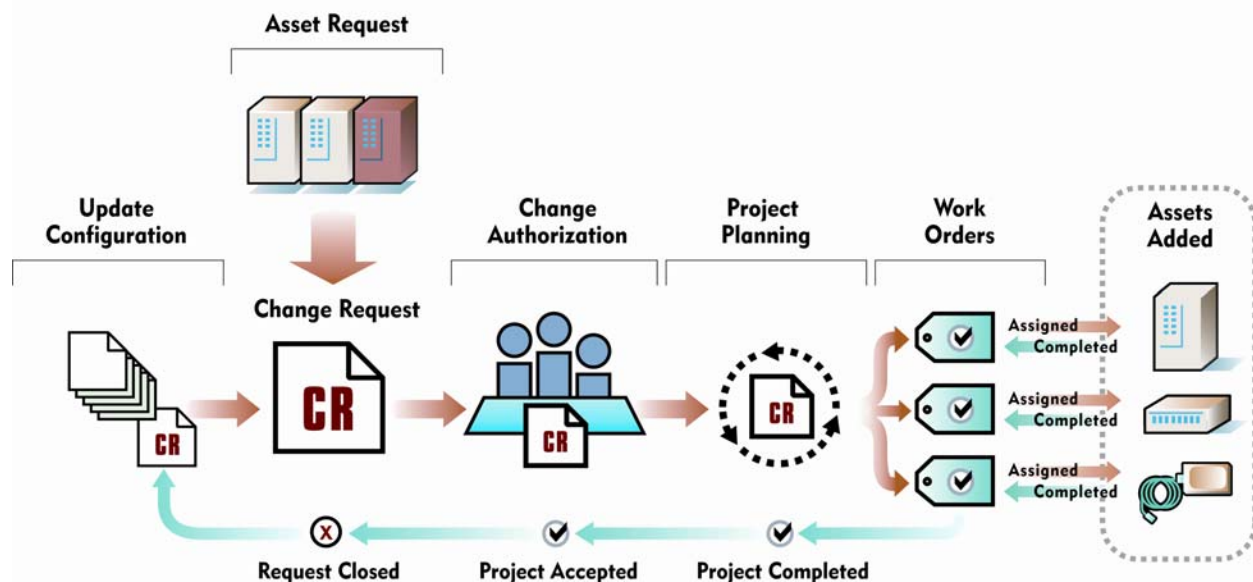


Figure 6.6: Visualizing complex workflows aids in user understanding.

The last information-retrieval technique is structured reporting. Traditionally, these reports support operational management. Reports serve several functions, including:

- Describing in detail the status of particular assets or configurations, including bill-of-materials.
- Identifying exceptions to a plan
- Summarizing the overall status of a set of assets
- Listing dependencies
- Identifying workflows that manipulate an asset

No one information retrieval and reporting tool can meet all enterprise needs. ECM tools should be flexible enough to allow the integrated use of multiple tools. Visualization could be used to quickly identify the main elements of an enterprise process, search and navigation can isolate particular elements within that process, and structured reports produce detailed descriptions of individual assets.

Functional requirements tend to focus on the intersection of end users and tools. Technical requirements shift the focus to the underlying implementation issues.

Understanding Technical Requirements and Options

ECM tools will function in the IT infrastructure of an organization. For most enterprises, that means a distributed, heterogeneous environment. To meet the demands of functional requirements while fitting into an existing environment, ECM tools should be judged on several technical requirements, particularly:


- Platforms supported
- Scalability
- Integration
- Systems administration

Work with Existing Platforms


IT infrastructures have evolved over the past three to four decades. It is not unusual to find latest-generation ERP systems hosted on state-of-the-art UNIX servers residing on the same network as 20 year old IDMS database applications running on mainframes designed a generation ago. ECM applications should work with the technology that currently exists within an organization. To do so, the tools must support multiple platforms, including, at least:

- Web
- Client/server
- Mainframe

Of course, each of these is a broad category representing multiple variations. Web platforms include HTTP, HTML and related scripting languages, and other protocols, such as Web Services protocols and WebDAV, which extends HTTP to allow users to manage files and collaborate over the Web.

 For more information about WebDAV, see <http://www.webdav.org>. You can also find technical information provided by the Internet Engineering Task Force (IETF) at <http://www.ietf.org/html.charters/webdav-charter.html>.

Web platforms must also include portals, which are frameworks for integrating applications and services in a Web environment. Applications are deployed in portals through components called *portlets*. Portlets vary from vendor to vendor, but several standards are under development that should eventually provide a common portal base for ECM developers and others.

 META Group's "The Emergence of Portlets" offers a high-level perspective on the future of portlet adoption and is available at http://itmanagement.earthweb.com/it_res/article.php/1146261. Technical details about Web Services for Remote Portlets (WSRP) are available at <http://www.oasis-open.org/committees/download.php/3343/oasis-200304-wsrp-specification-1.0.pdf>.

Client/server is a two-tiered model based on client machines running visual interfaces and business logic and servers providing database services. Windows, Macintosh, and UNIX-based systems can all serve as clients. Each of these requires different versions of client software. Web-based deployments offer more flexibility for both developers and users, but some functionality is still easier to deploy on client machines. Client/server is no longer the dominant architecture for distributed applications, but ECM vendors will need to support it along with other deployment schemes.

Mainframes continue to meet the high-performance, high-transaction needs of large enterprises. Although there was discussion about the demise of the mainframe during the heyday of the client/server architecture and the early days of the Web, the mainframe has continued as the backbone of large data centers. Today, companies are leveraging the high performance, security, and scalability of mainframes to deliver e-business services as well as traditional batch and basic transaction processing. Mainframes are vital to enterprises and ECM applications need to accommodate this platform along with others.

Scaling to Perform

ECM applications must be designed for high performance. These systems track low-level details about changes to code baselines, the results of builds, editorial changes to documents, routing logic in workflows, and other asset, dependency and workflow issues. Several design elements provide the needed scalability:

- High-performance relational database
- Designs optimized for the network environment
- Data model optimizations
- Pooling distributed resources

ECM applications generate large amounts of raw data that must be efficiently managed. The best tool for that job is a relational database management system (RDBMS). Relational database application designers and RDBMS vendors have developed many techniques to deliver rapid responses to large numbers of concurrent users. These databases also provide security and access controls for ECM applications. Major database vendors, such as Oracle, IBM, and Microsoft, have deployed distributed versions of their databases, which are useful from an ECM perspective as well.

ECM applications generate a lot of network traffic. Even simple changes to a document will create traffic to save the document and record the metadata about the change in a version-control system such as a document-management system. If the change triggers an event in a workflow, additional traffic is generated to send notification messages and route the document to the next party. These events have secondary effects, such as checks against access control databases, generation of audit information, and replication of changes to failover servers. Databases are tuned to optimize performance within the data storage realm. ECM core applications should be designed to minimize unnecessary network traffic. For example, they can cache data locally and to minimize redundant data retrieval.

Data storage in an ECM tool should be designed to read and write for high volumes of data. For example, to minimize the time required to search document metadata, the metadata should be stored separately from the documents. Metadata is generally much shorter than its corresponding document, so it can be stored in data structures optimized for small amounts of data. This scheme would not work for documents, however.

Documents can be stored either outside the database in a file system, in which case they do not incur the additional overhead found in RDBMS or in the database in a storage scheme designed for retrieving large blocks of data. Retrieving data from disk is relatively slow compared with other operations. Retrieving fewer but bigger chunks of data is one way to optimize large data access operations. The same optimization would not work for metadata retrieval because of the smaller size of the data. Additional unnecessary time would be spent accessing data.

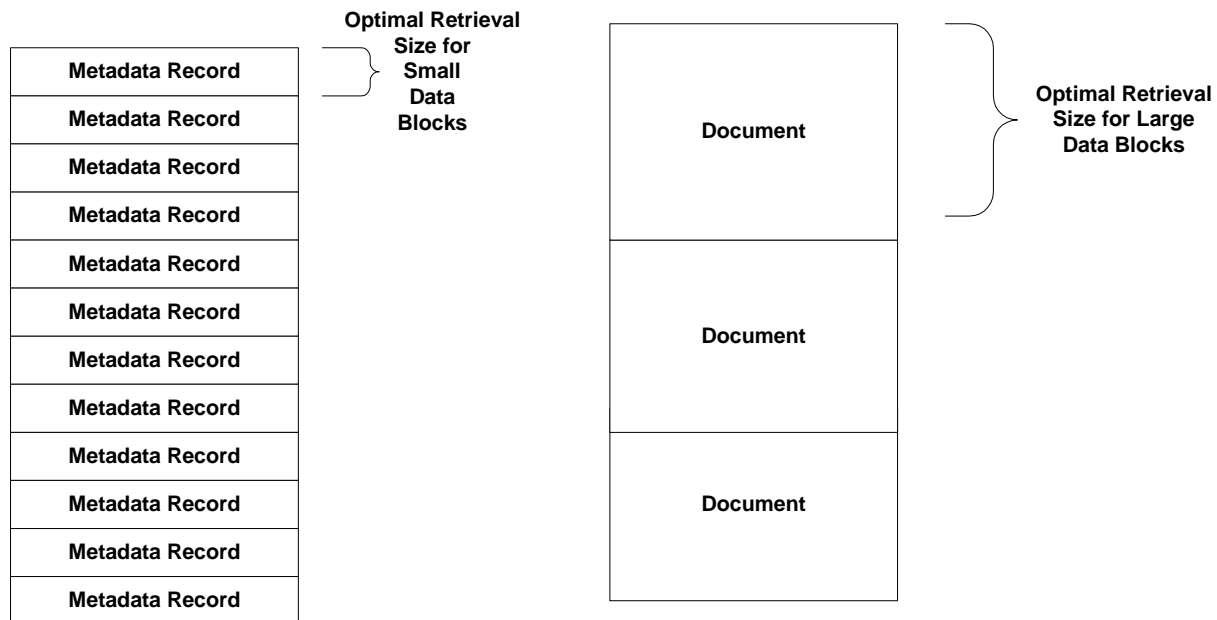


Figure 6.7: Data storage is allocated based on several factors, including average size of data records. Small records are stored with small allocations, or block sizes, while large records require larger data blocks.

When users interact with an application, the application often creates a process dedicated to that user. This design is simple and works well when the number of users is limited. As the number of users grows, the overhead involved in creating, managing, and deleting processes diminishes overall system performance.

One method for addressing this problem is to pool processes. Rather than create separate processes for each user, an application manages a set, or pool, of processes that are shared among users. This configuration reduces the demand on the server and improves overall performance. Additional performance gains are realized when the ECM application supports pooling.

Integrating ECM Components

One of the key advantages of an ECM system compared with silo-based change-management systems is the ability to analyze dependencies and workflows that cross traditional change-management boundaries. For example, a change in a policy for creating baselines in a software development project should trigger a change in both the source code version control system and the document-management system used by developers.

Reporting should also cross boundaries. If a server is scheduled for an upgrade, which applications are affected? Are any of those applications involved in critical workflows? Obvious workflows are easy to identify. An accounting system generates invoices and processes payments, but it might also provide reference data to an enterprise data warehouse that is updated nightly. To effectively meet ECM requirements, metadata about assets, dependencies, workflows, and policies require tight integration. Optimally, this integration is at the database level; less effective is interface-level integration.

Metadata integrated at the database level share a common data model in an integrated repository. ECM components are organized according to their type, such as assets and dependencies, rather than by their functional domain, such as software development and systems configuration management. The advantages of this approach are twofold. First, it provides an environment in which new domains can be added without fundamentally changing the ECM application. ECM might initially focus on software development, systems configuration management, and document management but other domains (for example, strategic planning and marketing campaigns) can also benefit from ECM practices.

Second, a consistent model allows for a single set of reports and analysis tools to work across functional domains, minimizing user training. More important, it allows for cross-domain analysis not available in silo-based approaches (see Figure 6.8).

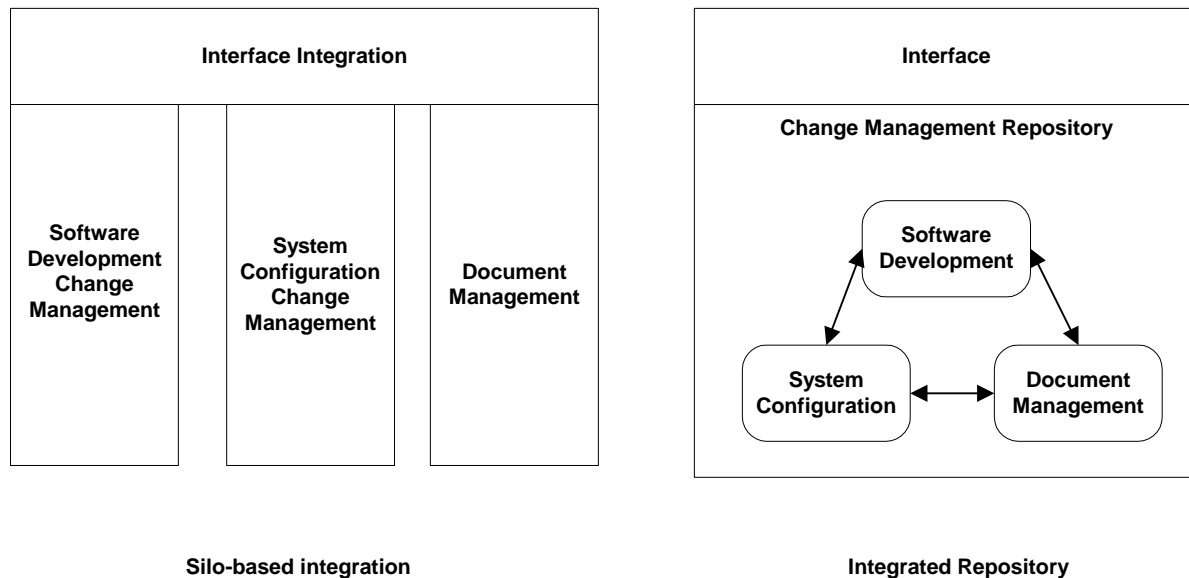


Figure 6.8: Integration at the data-model level allows for cross-domain analysis. Silo-based designs are challenged to perform similar analysis.

ECM applications should integrate with tools that support domain-specific functions. Software developers often use testing tools to generate test scripts, run suites of tests, and analyze the results. Content producers use desktop publishing, image, and video editing programs and other tools that need to function with document-management systems. When available, ECM tools should use open standards, such as Web Services protocols (for example, SOAP), to ensure the widest possible level of integration.

The capability to integrate data within an application and share it with other systems is a fundamental product of system design. ECM applications that will adapt to the diverse and changing needs of an organization are built on integrated repositories and open standards.

Administering ECM Applications

Another functional requirement to consider with ECM applications is systems administration. ECM administration factors include:

- Support for SSO
- Client integration with portals
- Integration with LDAP
- Backup and recovery procedures
- Server failover

- Support for clustered servers
- Support for an organization's standard RDBMS
- Routine maintenance procedures
- Enterprise application integration
- System integrity and security


Of course, these and other administrative issues underscore that fact that ECM applications are subject to the same systems configuration management issues as other enterprise applications. Platform support, scalability, integration, and systems administration issues are central to the technical requirements of ECM applications. The next section will address some of the functional requirements of these tools.

Estimating ROI of ECM Systems

The benefits of using an ECM system range from the obvious, such as saving software developers' time through version-control practices, to the obscure, such as preventing a fine from a government agency for not producing audit records that certify correct procedures were followed in a regulated process. Estimating the ROI of an ECM system requires an understanding of:

- Hard savings
- Soft savings
- TCO
- Regulatory requirements

The first three items are used in ROI and related calculations. The fourth, regulatory requirements, trumps the others. Regardless of the return or payback period, some industries need ECM to remain in compliance with regulations.

 For more information about the details of ROI and related calculations, such as Net Present Value, Internal Rate of Return, and Payback Periods, see "An ROI Primer" at <http://www.zdnet.com.au/newstech/enterprise/story/0,2000048640,20267881,00.htm> and "ROI Guide: Internal Rate of Return" at <http://www.computerworld.com/managementtopics/roi/story/0,10801,78524,00.html>.

Hard savings are definite reduced costs accountable to the use of an application. With regard to ECM applications, hard savings include:

- Eliminating software license and maintenance costs for silo-based systems, such as separate version-control and document-management systems
- Eliminating the overhead of maintaining ad hoc data interchange mechanisms for poorly integrated point solutions
- Consolidating servers and other hardware that had supported silo-based systems
- Reassigning support and administrative staff responsible for eliminated systems

Soft savings are harder to quantify. They do not entail tangible savings, such as a license cost, but typically tend to describe productivity gains, such as:

- Improved programmer productivity because of better access to requirements and dependency information
- Better project management through enforcement of policies, especially with regard to version control and baseline management
- Reduced time to create test suites, automate the execution of those suites, and distribute the results to appropriate developers
- Better identification of dependencies between network devices and fewer conflicts between components
- Less duplication of documents, fewer email attachments, and better control over official versions of documents

Cost is the third factor in the ROI equation. The term TCO has gained popularity to describe both the initial costs of acquiring an application, such as software licensing and server costs, and ongoing costs, such as maintenance and support. Again, ECM applications are like other enterprise-scale systems and incur costs beyond the initial expenditure, for example:

- Staff support including database administrators, network administrators, and Help desk personnel
- Staff training
- Maintenance contracts
- Hosting and infrastructure costs

ECM applications can extend the benefits of traditional change-management systems. Success is dependent on matching the needs of an organization with the appropriate application, deploying the application in a cost effective manner, and introducing change-management practices in ways that maximize user adoption. The first step in that processes is to understand the options available to you.

Assessing the ECM Market

ECM products are evolving from multiple points in the IT market. Some have a software-development orientation, others are more focused on document management, and still others approach the problem from a systems configuration perspective. In some cases, vendors offer near-ECM systems such as Product Lifecycle Management (PLM) applications.

The diversity of approaches underscores the complexity and breadth of ECM. It also demonstrates that there is no “domain free” perspective on ECM. Products in this space evolve from earlier change-management products. One way to assess these products is to examine:

- The domain or domains that orient the product
- The degree to which they can represent assets, dependencies, and workflows outside those original domains

These two issues are the subject of the next sections.

Approaching ECM from a Software Perspective

Consider, for example, Merant Dimensions. This ECM application has strong support for software configuration management (SCM), including baseline management, build management, and release management. Dimensions integrates with third-party SCM tools as well as other specialized tools such as Merant Mover for deploying applications across the enterprise and Mercury Test Detector/Defect Tracker for supporting quality assurance procedures. Although the roots of Dimensions are clearly in the SCM arena, the architecture is flexible enough to support other areas of ECM.

Dimensions tracks assets, dependencies, and workflows. Policies and roles ensure that assets are managed properly according to an organization's needs. Equally important, information about change-management objects are stored in a single repository ensuring the broad integration of assets that is demanded in ECM environments.

 For more information about Merant Dimensions, see <http://www.merant.com/Products/ECM/dimensions/home.asp>.

Products will vary in their support for non-software assets, such as documents, and their ties to particular methodologies. Support for methodologies can enhance software development. However, they should not be so fundamental to a tool that they hinder workflows or asset models outside software development.

Approaching ECM from a Document-Management Perspective

ECM applications can also have a document-management foundation. These systems expand beyond conventional document management to support the concept of enterprise document of record. This is more of a logical concept than a single physical document. For example, the enterprise document of record for a new product might consist of content stored in an ERP system, a content-management system, a PLM system as well as a CRM system. Supporting enterprise document of records in manufacturing environments improves collaborative engineering, product development, resource scheduling, and production operations.

Another example is the concept of a patient record. Patient information is distributed across a range of systems in contemporary healthcare systems, including clinical records systems, pharmaceutical systems, and image management systems for X-ray, CT scans, and MRIs. Under HIPAA legislation in the United States, much of this information is considered protected health information and is subject to strict regulations. Effectively managing this type of distributed "record" requires an enterprise-management perspective like that found in ECM.

Approaching ECM from a Systems-Configuration Perspective

Vendors can also tackle ECM from a base in systems configuration management. This type of change management starts with a focus on hardware, software, and network assets as well as on systems configurations and monitoring and dependencies between systems. These tools have strong support for automatic discovery of server state, storage usage, processor load, and other operational measures.

Some vendors offer specialized change-management tools that point to new development areas in ECM. These products do not fall into the ECM category because they are silo-based approaches, but they do provide functionality that ECM vendors will have to eventually support. For example, database change-management tools identify dependencies of proposed changes and evaluate the impact of changes on tables, views, stored procedures, and other database objects. They might also include support for migrating and converting data as necessary to implement changes.

Understanding the Current State of the Market

ECM is a new and evolving technology. Its roots are in specialized change-management domains such as software development, document management, and systems configuration management. Tools in this market have evolved from strong positions in at least one of these domains. In some cases, such as with Merant Dimensions, there are strong roots in both software development and configuration management.

As ECM continues to develop, we can expect vendors to improve support for the change-management areas outside their original focus. A key differentiator will be their approach. If vendors choose a unified, single repository approach with data models designed for high-level, cross-functional entities (such as assets and dependencies), the tools will maximize the benefit to users. If silo-based applications are collected in a single interface without the deep integration provided by a single repository and common data model, users will not realize the full benefits of ECM.

Introducing ECM to an Organization

Introducing ECM to an organization requires preparation. The first step is assessing the organization's readiness for ECM. This step is followed by a series of steps to introduce the concepts of ECM and determine the best approaches to integrating ECM with ongoing operations.

Assessing the Readiness for ECM

Realizing the benefits of ECM depends on users adopting ECM practices as part of their day-to-day activities. Many staff, contractors, and consultants are already under pressure to produce more with less, meet tight deadlines, and take on new responsibilities. To understand better how ECM will be accepted, consider several questions.

Are Change Management Practices Already in Place?

Many software developers already use version control, testing, and quality control systems. Extending these practices to include better document management and systems configuration considerations should not be difficult. If basic version-control practices are not used, even if the systems are available, adoption will be more difficult.

If basic change-management systems have become “shelfware,” determine what caused the poor acceptance. Does the tool not support effective workflow? Is the interface difficult to use? Does the tool only solve part of a problem? If these are the cause of the poor adoption, ECM could provide the solution.

Look for similar situations with document management and systems configuration management. Are the repositories up to date with the latest documents and components? Is the metadata associated with assets complete and useful? Users might choose not to add metadata when saving and modifying documents. This shortcoming indicates that users are not aware of how useful the metadata actually is and probably do not know how to use the additional information for information retrieval. The repositories have become archives rather than operational support tools. Again ECM may address these issues. If a tool fits the way staff works and developers, content providers, and network administrators know that information they provide is actually used to improve operations, the system is more likely to be adopted.

Is There Executive Sponsorship for ECM?

Executive sponsorship is a key factor in organizational readiness. Sponsorship cannot stop at approving a budget; it must follow through into the deployment and adoption of ECM. Executive sponsors should:

- Define the role and benefits of ECM in the organization
- Outline a long-term plan for adopting ECM in an incremental fashion
- Understand the risk factors, such as potential migration and integration problems
- Define management practices based on ECM
- Identify ways ECM tools are used to ensure compliance

The greatest benefits will be realized with ECM if it is used across departments, projects, and subject areas. Executive sponsorship is essential to meeting that objective.

What Problems Are Solved by ECM?

It is important to understand the drivers behind ECM use in an organization. Some may be clear (for example, companies need ECM to remain in compliance with government regulations). In other cases, users of change-management systems already understand the benefits of the discipline and want to extend its reach. ECM technology can address a variety of issues. One source of information about such issues is post-project assessments. Review these assessments with a particular eye to understanding:

- Communication problems within the team
- Code-management problems
- Unusually long time to resolve bugs and requirements issues
- Failure to implement required functions
- Late discovery of integration problems
- Poor communication on overall progress of the project
- Executive sponsorship
- Introduction of ECM in response to poor performance, compliance

The benefits of ECM are not limited to projects.

Are Daily Operations Constrained as the Result of a Lack of ECM?

Infrastructure management can be especially difficult without centralized change management. Tell-tale signs of the need for ECM from a systems configuration management perspective include:

- Projects that are hindered because of a “don’t change the infrastructure, something might break” attitude
- Network administrators using multiple tools for monitoring system performance, configuration, and status but without the ability to integrate the information gathered from these systems
- Lack of any formal change-approval process result in authorized production downtime
- Inability to report on the status and configuration of servers and other hardware used by a particular application

Assessing an organization’s readiness for ECM entails understanding past experience with change management, operational issues that will benefit from ECM, and the level of executive leaderships prompting the project. When it is clear that there is a need and support for ECM, it is time to develop a deployment strategy.

Deploying ECM

There is no magic formula for deploying ECM, but you can follow some general guidelines:

- Start with small groups and deploy incrementally. Ideally, a rollout starts with teams or departments already familiar with change management. The initial focus with these groups would be to extend the scope of their change-management practices. For example, having software developers use document management and system configuration management along with code-management features.
- Start with midsized projects that need ECM but are not overly constrained by tight deadlines or budgets. When the pressure is on, teams have a tendency to abandon new practices and resort to familiar habits.
- Find cross-departmental projects that require coordination. ECM can be especially useful with distributed teams.
- Set expectations that project managers will manage from ECM content. Make the system the official record of source.

Organizations will adopt ECM at different rates and for different benefits. Those that succeed will start with a clear understanding of the business objectives and a realistic deployment plan based on an incremental deployment.

Benefits of ECM

ECM introduces several types of benefits to organizations, including:

- Better alignment of IT operations and business needs
- Improved IT quality control
- Improved ROI on software development and management
- Improved infrastructure management and security
- More predictable project life cycles

Aligning IT with Business

Managers can better understand business needs and IT constraints when information about both is readily available. Single repository change-management systems provide reporting on technical aspects of a project, such as the code development life cycle, along with information about the sometimes dynamic business requirements. ECM provides the basis for a single, holistic view of a project, not just domain-specific aspects of it.

Improving IT Quality Control

ECM also improves on IT quality control, especially when ECM applications integrate with testing and deployment tools. Merant Dimensions, for example, works with tools for building applications from source code, executing test suites, and deploying applications to production environments.

Improving ROI on Development

As noted earlier, ECM practices can improve factors that influence project ROI. The practices have a direct effect on reducing lost changes to code, poor version management, and policy enforcement. Managing to requirements, effective bug tracking, and providing transparent reporting on project status indirectly contribute to project success rates.

Improving Infrastructure Management

When software, business process, and systems configuration management information is integrated, organizations can better manage the dependencies between assets. Improved infrastructure management will lead to fewer unanticipated consequences when assets change.

Predicting Project Life Cycles

Finally, ECM applications track detailed information about the course of project, the life cycle of assets, and the dynamics of organizational infrastructure. This data becomes a foundation for understanding typical project life cycles and daily operations in the organization. More data leads to better analysis, and better analysis supports better management.

The Future of ECM

The fundamental elements of ECM, assets, dependencies, workflows, policies, and roles will not change with time. Objects of change management and the breadth of ECM will be the focus of ECM evolution. To complete this guide, let's explore the future of ECM.

Evolving Objects of ECM

The objects of change management are evolving even as ECM emerging. Software development is adopting the new paradigm of Web services. Records management is no longer limited to information in a single database but span systems within an organization and beyond.

Dynamics of Web Services

In the case of software development, Web services present new problems to change management. Traditional mainframe and client/server development assumed relatively controlled execution environments. Applications were fairly self-contained and made limited use of OS services and code libraries. If there was a problem with a build, it could usually be traced back to a problem with the application source code. Web services are changing that model.

Applications designed around Web services invoke programs on other servers, inside and outside the firewall boundary of the organization. Web services use common protocols for defining services, directories for finding services, and exchanging messages. Keeping provider services in sync with consumer services is a challenge because they can change independently of each other. In many cases, this is one of the appealing features of a Web service. For example, if you have an e-commerce operation, you might subscribe to a tax calculation Web service. The service calculates state and local taxes on customer purchases. Tax rules and rates can change without any changes to your application, only the tax Web service needs to change. Consumer applications of the service are unaffected. If a change in tax code requires more information to calculate tax (for example, an indication of whether the product is clothing), the consumer service will need to change the information sent to the provider service. Detecting and managing changes in large numbers of Web services will present new challenges to ECM applications.

Composite Objects and Logical Records Management

Logically related data is becoming highly distributed. Patient healthcare records and financial services customer records are just two examples of regulated information. The fact that each of these is composed of multiple records, or assets, and must be managed as a logical unit will force advancement in ECM models.

We cannot think of assets as autonomous units, such as a source code file, a word processing document, or a row in a relational database table. Assets are composite objects. Each component is linked to the others but has a distinct life cycle. Separate versions may exist and be considered valid simultaneously.

When a patient record is transferred from one doctor to another, it might include only part of the original logical record. There should be some indication with the original that a segment of the record was sent to another location. That new version may combine with segments from earlier versions of the same patient's information. Tracking the composition process of logical records and managing the effects of independent life cycles are especially challenging.

Evolving Processes of ECM

ECM encompasses traditional change-management domains: software development, document management, and systems configuration management. Other business processes will be quickly drawn into the fold:

- Business process re-engineering
- Research and development
- Product development
- Manufacturing processes

Successfully expanding the scope of ECM will depend on new tools as well as better integration.

The Evolving ECM Toolset

As ECM models evolve, we can expect metadata standards to emerge. These standards should provide a basis for describing abstract objects that are shared across ECM tools and components. The complexity and breadth of ECM repositories will also trigger the development of search and navigation tools, especially visualization applications to aid ECM administrators and users. Analytic operations, such as link analysis for detecting wide-spread dependencies, will also emerge.

ECM is the evolution and merging of practices that have proven their value for decades. Like its predecessors, ECM will meet the immediate needs of organizations while remaining flexible and adaptive to the dynamic nature of today's enterprises.