Unified Change Management from Rational Software:

An Activity-Based Process for Managing Change

Rational Software White Paper

speed quality

Rational®
the e-development company™

Table of Contents

Introduction

Rational® Software offers a reliable and comprehensive change management solution that integrates the capabilities of two leading tools: Rational ClearCase® for software configuration management and Rational ClearQuest® for defect and change tracking.

Both Rational ClearCase and Rational ClearQuest provide extensive flexibility for configuring processes that manage change in software development. When used together, these powerful tools simplify change management by clearly identifying associations between any type of change request — whether enhancement request or bug fix — with related changes in application code or Web content.

Unified Change Management takes the integration of software configuration management and defect and change tracking several steps further by providing a pre-defined process that organizes work around activities and artifacts. Based on the Rational best practice of managing change, Unified Change Management is enabled by Rational ClearCase and Rational ClearQuest. It accelerates development with an out-of-the-box process model that can be turned on or off, based on the specific needs of a software development team.

Included in the Rational Suite™ integrated toolset, Unified Change Management simplifies change management across the software development lifecycle by automating the management of all types of software development artifacts. These can include requirements documents, design models and testing artifacts, in addition to application code and HTML and XML content.

Unified Change Management From Rational Software: An Activity-Based Process for Managing Change, describes Unified Change Management in detail, its advantages and how teams can benefit from it with Rational ClearCase, Rational ClearQuest and Rational Suite.

Change in the Software Development Process

Market Drivers

Software teams today face significant challenges. The demands of an Internet-driven economy require the production of high quality software applications at unprecedented speeds. Software application requirements continue to grow more complex, along with the structure of the development environment. Distributed development teams, heightened performance demands, multiple platforms, shorter and continuous release cycles — these factors and others intensify the traditional pressures of a software development environment. Add to these challenges the escalating cost of failure. Companies often “bet the business” on the success of a new application. In a business environment where technology serves as organizations’ primary interface with customers, application successes and failures are highly visible and costly.

Team Dynamics

“Developing quality software in a repeatable and predictable fashion is a job that requires the coordinated activity of a team of developers. Although individual productivity is still important as systems get larger and more complex, the productivity of the team as a whole becomes a much more important factor in the success or failure of a project.”[1]

The productivity of a software development team is largely determined by its ability to coordinate and organize activities. And its success is strongly linked to how efficiently it can respond to changing environmental factors. Internal activities create constant change in the software development environment in the form of new feature introductions, new enhancement requests and defect fixes. External factors — such as new operating environments, the integration of new tools, improvements in engineering techniques and market conditions — drive change with additional force.

Resistance to change is not a viable solution for development organizations that need to compete in Internet time. Nor is straying off course to respond immediately to every change request.

[1] “The Software Development Team White paper”, Grady Booch, Rational Software, Corporation, January 1999.

Successful software development teams adopt strategies to deal with the inevitability of change. These change management strategies can be based on experience, preferences and organizational mandates. And they often stem from manual processes that evolve with organizational growth and changing requirements. A critical success factor for any change management process is its ability to scale.

Unified Change Management

As development organizations grow and release cycles accelerate, the need for tools and processes that streamline the management of software change becomes essential. Rational Software's Unified Change Management (UCM) delivers this process with a tooling infrastructure provided by Rational ClearCase, Rational ClearQuest and Rational Suite. Powered by Rational ClearCase and Rational ClearQuest, UCM is Rational's activity-based process for change management.

Activities and Artifacts

As software systems and teams grow in size and complexity, it becomes increasingly important for teams to logically organize activities around periodic releases and efficiently manage the artifacts used to implement these activities. An *activity* can involve fixing a defect or building an enhancement for an existing product. An *artifact* is anything that evolves over the course of the development lifecycle, such as a requirements document, source code, design model or test script. Teams perform *activities* and produce *artifacts* (Figure 1). UCM integrates the activity management capabilities provided by Rational ClearQuest with the artifact management functions of Rational ClearCase.

ACTIVITIES
ACTIVITY
ACTIVITY
ACTIVITY
ARTIFACTS

Figure 1: Unified Change Management integrates Rational ClearQuest's activity management capabilities with the artifact management functions of Rational ClearCase

Activity Management

Activity management in UCM is enabled by Rational ClearQuest — a highly flexible and scalable defect and change tracking system that captures and tracks all types of change requests — such as defects and enhancement requests. In UCM, these changes are viewed as activities.

Rational ClearQuest offers a customizable workflow for tracking and managing activities. This enables teams to more easily:

- Assign activities to specific individuals
- Identify priorities, status and other information associated with activities
- Automatically generate queries, reports and charts

The ClearQuest workflow engine can be adapted based on team or procedural requirements. Teams without customization needs or a predefined change management process can use ClearQuest's out-of-the-box process, forms and associated rules for fast implementation (Figure 2). While teams requiring extensive customization can use Rational ClearQuest to tailor just about every aspect of their change management system – including defect and change request state transition lifecycles, database fields, user interface layouts, reports, charts and queries.

An effective workflow for managing and tracking defects and other changes throughout the development process is essential for meeting today's high quality standards and tight production deadlines. UCM raises the level of abstraction of these changes to view them in terms of activities. It then leverages the Rational ClearQuest workflow engine to link activity management to associated development artifacts.

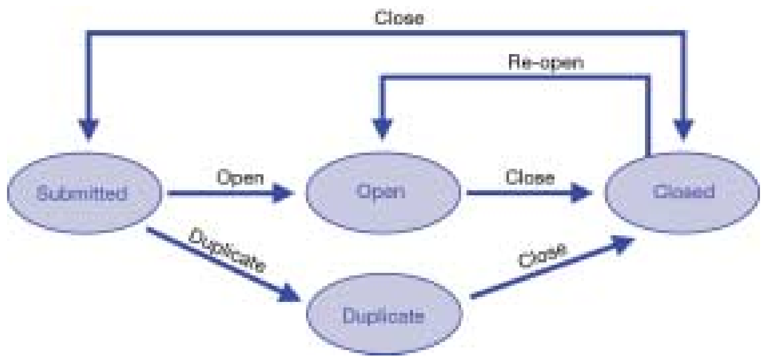


Figure 2: Rational ClearQuest provides an out-of-the-box process framework for defect and change tracking workflow

Artifact Management

Rational ClearCase provides the software configuration management (SCM) infrastructure that UCM leverages to automate management of artifact development across the project lifecycle. Rational ClearCase is the market leading SCM tool[2] that offers:

- Secure digital artifact storage and versioning
- A parallel development infrastructure — unlimited branching capabilities coupled with sophisticated merge functions
- Build management
- Workspace management for selection of the correct versions of artifacts
- Complete scalability from small, co-located project workgroups to large, globally distributed teams

In addition, Rational ClearCase provides a flexible infrastructure for implementing SCM. With embedded metadata, triggers and hyperlinks, teams can use ClearCase to implement custom SCM processes based on the level of process enforcement required.

For example, to meet auditing and maintenance requirements, teams can attach *comments* metadata on check-in. They can enforce policies that manage the use of these comments to insure that they exist, meet minimum length standards and are spelled accurately. Other teams may find it sufficient to simply 'recommend' policies for completing comments.

With Rational ClearCase, organizations benefit from the flexibility of using different policies for different teams or projects. With UCM, they get an out-of-the-box process that automatically implements these project policies based on proven, successful development processes.

[2] Application Life-Cycle Management Forecast and Analysis, 2000-2004. 2000 IDC

UCM: Five Process Areas

UCM provides defined processes for five specific areas:

- Developer insulation and collaboration of shared and common code artifacts
- Collecting groups of related artifacts that are developed, integrated and released together into UCM components
- Creating component baselines at project milestones and promoting baselines *as complete sets* according to established quality standards
- Organization of changes into change sets
- Integrating activity and artifact management

Developer Insulation and Collaboration

Developers require insulated work environments to isolate their work-in-progress from the potentially destabilizing changes of other team members. Rational ClearCase provides two options for accessing the correct versions of artifacts and working on them in private workspaces. These include *snapshot views* and *dynamic views*, which are optimized, respectively, for either local or networked use.

Snapshot views provide team members with the flexibility to work disconnected from the network and easily synchronize their work with mainline development. Dynamic views are implemented with a unique virtual file system that enables team members to transparently access the right versions of the right artifacts, without copying them to their hard drive. Regardless of how they're created, these private work areas allow team members to work simultaneously on multiple releases.

For example, a developer working on release 2 can also fix a defect in release 1, without fear of altering code that's unrelated to the developer's assigned activities. In another example, one developer can fix a defect in release 2 while another works on a new feature addition. Because Rational ClearCase enables each developer to perform their respective activities in their own private work areas, neither individual need worry that their work will be impacted by the other's changes.

While insulation from destabilizing changes is critical for minimizing error, the integration of all changes into a common work area that's accessible to all team members is an essential step in team development. Today's component-based development methodologies, along with the increasing frequency and pace of code changes, require teams to integrate work often.

With Rational ClearCase, teams can implement a variety of project policies to insulate and enable collaboration, simultaneously. For example, a small team may use a serial methodology allowing just one developer at a time to make changes to an artifact. In contrast, a policy like this would hinder the productivity of large teams requiring multiple developers to collaborate on common artifacts.

Rational ClearCase supports parallel development on a large scale with rich branching and merging capabilities. In UCM, the branches enabling parallel development are viewed at a more abstract level as *streams*. A stream represents a workspace that can be private or shared. Streams define consistent configurations of project versions and provide the balance between isolation and sharing that enables efficient collaboration in a UCM project. To achieve this balance, Rational ClearCase configures private streams for individual team members along with public integration streams where all work is delivered (Figure 3).

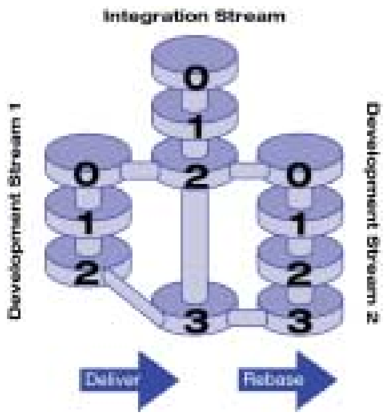


Figure 3: UCM provides public integration and private work areas

Private streams provide developers with insulated workspaces initialized with the public artifacts that meet established quality standards. Developers use these private workspaces to make changes, and build and test artifacts. When developers are satisfied with their changes, they deliver them to the public integration stream. To keep up to date on project changes made by other team members, developers can also *rebase* their private work streams with the most recent, stable baselines from the project's public integration stream. With UCM, developers choose when to deliver and rebase.

The public integration streams serve as the coordinating point for all project changes made by all team members. UCM introduces the concept of a *baseline* as a measure of progress. A baseline, defined as an abstract representation of a build or configuration, is a version of a *component*. A component is a collection of related artifacts. Project teams create and promote baselines throughout the development process. As different developers deliver changes to the integration stream, they are collected into a component baseline. As baselines are built, tested and approved, they are promoted through various baseline levels.

Baseline promotion levels provide two things. First, they enable managers to establish software quality standards. Managers can set project policies that identify the baseline levels at which developers can perform rebase operations. These baseline levels are reached when the baseline achieves a predefined measure of quality. Second, baseline promotion levels provide guidance on how specific team members should interact with artifacts under development. For example, they help testers determine when to begin testing, based on when a baseline passes certain smoke tests.

Component Baselines

The second UCM process guides the organization of artifacts into components. SCM systems provide an infrastructure for managing versions of digital artifacts, such as files containing source code. Teams can achieve significant gains in productivity with these systems, which eliminate the need to manually track artifact versions.

UCM expands on SCM versioning capabilities by consolidating individual artifacts into components and providing tools and processes for automating component management. Components are baselined as a unit, which can then be used as the foundation for new development and to update developer workspaces.

The automation of component management is critical for the efficient and error-free development of complex software systems that can contain thousands of source code artifacts — in addition to a host of other related artifacts — such as content, design models, requirements specifications and test scripts.

Component Baseline Promotion

Members of a project team work in a UCM *project*. Projects promote component architectures by enabling managers to configure software components. Most organizations design UCM-managed components to reflect their software architecture (Figure 4), organizing all the artifacts comprising architecturally significant subsystems into individual components.

Projects are defined by the components they work with, including any relationships between the project and its components. For example, a UCM project can specify whether team members are authorized to modify a specific component or simply reference it.

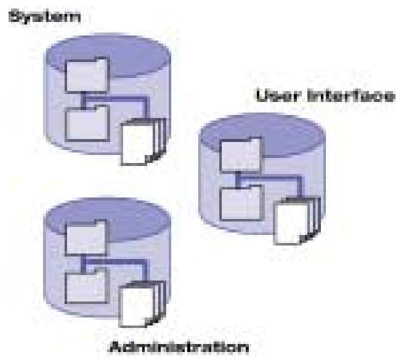


Figure 4: UCM components directly model software architecture

As described in a previous section, developers periodically update the components in their private streams when they deliver changes to the public integration stream. Teams can then rank components according to stage in the development process and quality level. Project policies determine the quality level a component baseline must reach before developers can rebase and how other team members, such as testers, should interact with the component baseline.

UCM provides five, out-of-the-box baseline levels. These include rejected, initial, built, tested and released. In addition, UCM enables teams to customize these baseline levels with their own naming conventions and promotion policies.

Change Sets

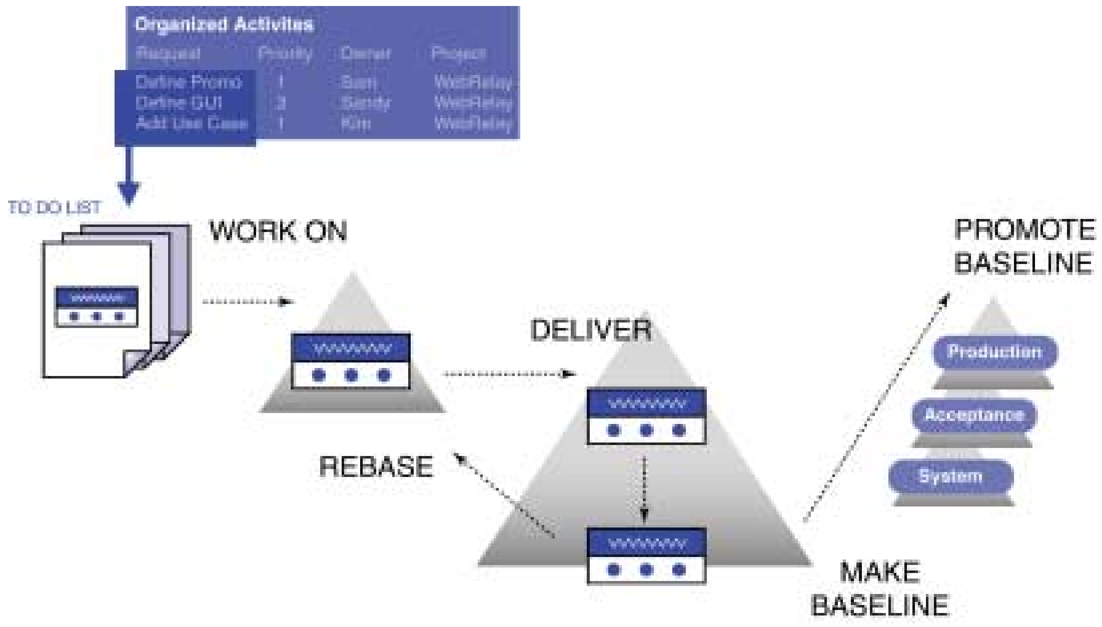
The fourth UCM process involves organizing individual artifact changes into *change sets* that are delivered, tracked and managed as a whole. A change set represents all artifacts changed to complete a specific activity. When developers work on an activity, such as a defect fix, they rarely change only one file. In most cases, individual file changes are useful only when organized and managed in conjunction with all the changes made to the artifacts required to implement that change.

The need for developers to work on many changes at one time complicates the process even more. Consider, for example, the developer who needs to interrupt work on a new release to fix a defect in the current release. Chances are, the developer must make multiple changes to the same artifact. These may include changes in progress for the enhancement and others to fix the bug in the previous release.

UCM simplifies the process of managing multiple changes to artifacts by collecting all changes for an activity into a change set. UCM organizes work around specific activities. It also ensures that a completed activity incorporates all the changes made to all the appropriate artifacts.

Activity and Artifact Management

The fifth UCM process integrates activity and artifact management with an automated workflow that links activities to associated sets of artifacts (Figure 5). This gives teams the flexibility to specify different pre-defined workflows for managing different types of activities. UCM provides predefined processes for the most common types of activities, including defect fixes and enhancement requests — right out of the box.



Organized Activities

Request	Priority	Owner	Project
Define Promo	1	Sam	WebRelay
Define GUI	3	Sandy	WebRelay
Add Use Case	1	Kim	WebRelay

Figure 5: Overview of a UCM workflow

Teams can also customize these processes with the *ClearQuest Designer* module that enables a manager to create all required activity types. These can include defect fixes and enhancement requests, along with documentation changes or Web site modifications. With the ClearCase Designer's graphic user interface, managers can also define the fields, forms and lifecycles for each record type.

To enable team members to more easily identify the relationships between activities and the artifacts comprising a project's code base, UCM automatically links activities to their associated change sets (Figure 6). When working in a UCM project, developers deliver activities (see Developer Insulation and Collaboration), rather than individual file artifacts. Similarly, when developers rebase, they can review what they will receive in their private streams based on the activities the new component baseline provides.

Integrating activity and artifact management enables teams to populate private and public streams with much more than just the current version of a project's artifacts. With UCM, team members can view the complete version history of all related artifacts, along with all the activities that implemented each change. This gives teams a comprehensive picture of how a project evolved from one stage to the next – an invaluable advantage when the need arises to identify how changes made to one artifact version have impacted another.

With UCM's consistent, repeatable processes for activity management, teams reduce error by adhering to automatically applied review cycles. They also work more efficiently by bypassing many of the tedious, manual steps typically required to perform development tasks.

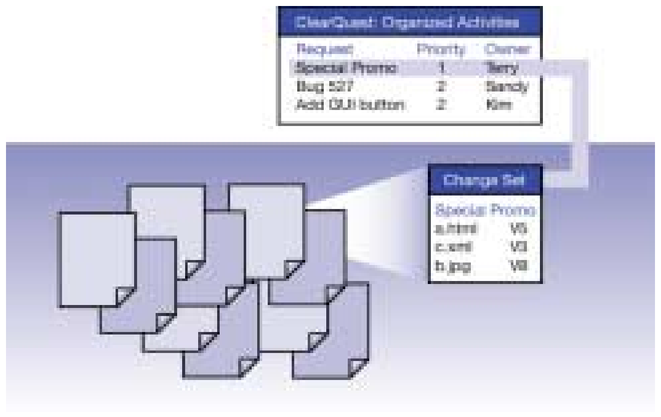


Figure 6: UCM links activities to associated artifacts

Team Adoption

Successful change management is as much about people's activities as it is about appropriate tooling. A change management infrastructure will fail if team members can't or won't adopt its methodology.

Rational Software drew from the "best practices" of successful software teams and its own extensive software development experience to design UCM as a process optimized for team adoption. UCM capabilities are fully incorporated in the Rational Unified Process™ (RUP), Rational's practical framework for applying best software development practices across the lifecycle. Software teams have immediate access to detailed descriptions and UCM process direction by accessing RUP directly and through tool mentors in Rational ClearCase, Rational ClearQuest and Rational Suite.

UCM simplifies work for developers by enabling them to:

- Access the right versions of the right artifacts, quickly and accurately
- Work in isolation, and choose when to deliver work and receive others' changes in their workspace
- Use project policy to identify when quality has reached an acceptable level for receiving changes into their workspaces
- Access change management tools transparently from their desktops. Use tools that integrate with the tools they use most — without undue constraints on their activities or increasing the time it takes to do their job
- Automatically and transparently integrate change sets with activities
- Automatically track, manage and report activities

Likewise, UCM eases administrative burdens for project managers with advantages that enable them to:

- Identify the status of a specific activity at any time with a prescribed lifecycle workflow
- Monitor up-to-the-minute project status with consolidated reporting
- Focus activities on high-priority or urgent events
- Determine workloads at a glance and balance assignments

Artifacts Across the Lifecycle

Thus far, this paper has focused predominantly on source code and its related artifacts, such as object files and headers, as well as developer interaction with UCM. The full benefits of Unified Change Management are best realized when it is used to manage change across the lifecycle by team members who are not necessarily developers. These can include analysts, designers and testers who produce artifacts relevant to their specific domains (Figure 7).

For example, analysts produce requirements documents and use cases. Designers also produce use cases, along with design models. Testers produce test scripts, test data and results.

To communicate and coordinate work effectively, team members need to be able to share these artifacts as efficiently as they share code. Rational Suite includes an integrated platform that enables common access to all types of development artifacts across multiple domains. As part of Rational Suite, UCM provides the process layer for managing the sharing of information across the software development lifecycle. Rational ClearQuest and Rational ClearCase enable the UCM process as part of the *Team Unifying Platform* included in every Rational Suite.

The same UCM principles used to control code changes are applied to the management of artifacts produced by team members who are not software developers. The Rational ClearQuest workflow engine powers activity management. A consistent workflow makes it easy for all team members to identify priorities and next steps with an intuitive *to do list* feature that can be transparently accessed from each individual's desktop. Component baselines serve as the foundation for new development and guide when team members can update their workspaces – providing all team members with the same level of artifact management UCM delivers to developers.

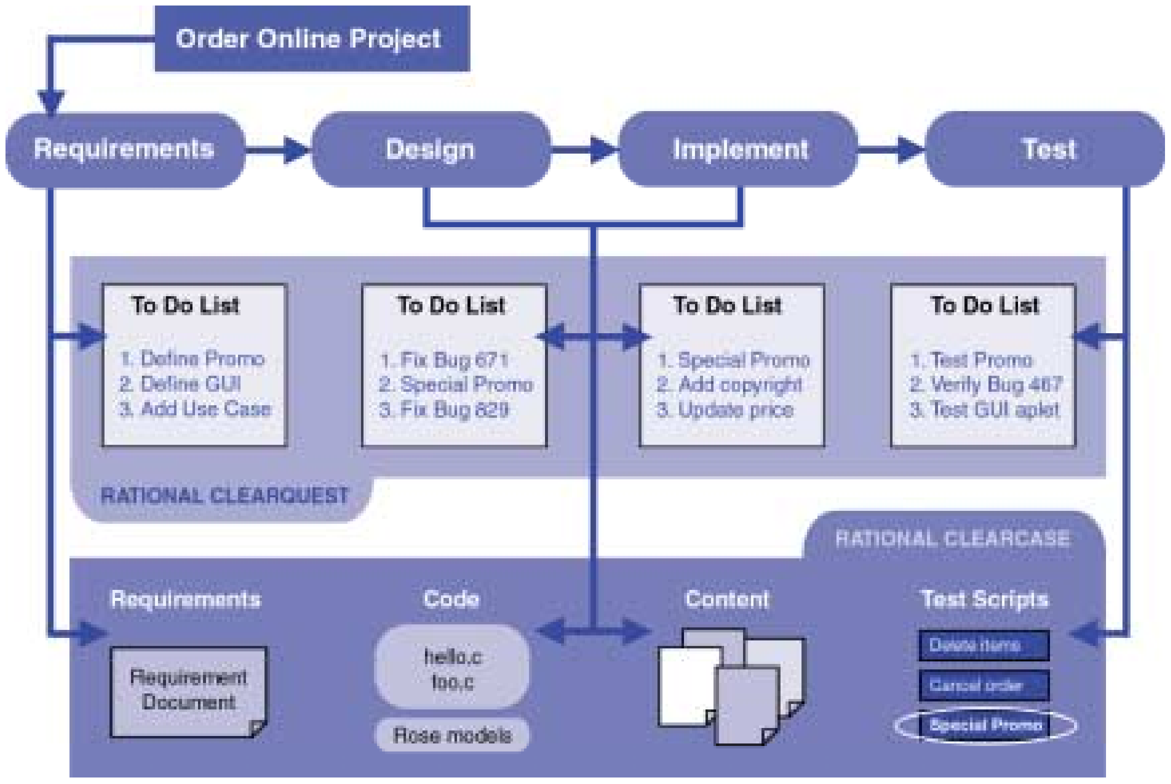


Figure 7: Team members generate a variety of different artifacts across the lifecycle

Artifacts from Analysis

Analysts perform the important role of defining project scope, which requires determining what the solution will do and its boundaries. During analysis, these professionals create a variety of different artifacts that help flesh out the proposed solution including requirements documents, use cases and visual models. It's essential that teams continue to use these artifacts to manage changes throughout the development process.

Successful projects rely on appropriate requirements management. Effective requirements management involves resisting changes that impose undue schedule risk or instability, and tracing requirements changes. Project management, risk assessment and metrics generation all rely on requirements management and traceability — the ability of the team to accept changes in a formal process and review the history of requirements changes.

Rational RequisitePro® is the enabling tool for requirements management in the Rational Suite Team Unifying Platform. Software teams use RequisitePro to create, manage and track analysis artifacts — such as requirements attributes, specifications and management plans — as well as use case models, glossaries and stakeholder requests.

A UCM project manages requirements similarly. It also includes these requirements artifacts in component baselines. As important as it is to know which activities make up the latest artifacts in a component baseline, it's also important to know which requirements guided the development of these artifacts.

UCM provides the link between the current state of the requirements and specific component baselines. UCM-enabling a RequisitePro requirements definition means similarly baselining these requirements hierarchies. This provides the critical facility to reconstruct any component baseline, in terms of its code artifacts as well as current project requirements. In addition, UCM allows team members to use RequisitePro to efficiently track the progress of requirements — making it easy to identify key information such as which requirements have been met and the changes made to implement them.

Artifacts from Design

Designers model the system architecture and further shape the system definition in terms of use cases and design models. These artifacts reduce the complexity of the overall system through abstraction. The early efforts of the design team will change considerably by the time the effort reaches the coding stage, both in terms of its level of detail and often its structure.

Some teams fail to continue employing these design models as the formal coding effort commences. This is a mistake, as these design models serve a valuable function in helping to shape the overall effort, assisting new team members get up to speed quickly, measuring the impact of ongoing change requests and assessing overall project risk. The round-trip engineering facilities provided by Rational Rose® help teams make and keep these models current as coding progresses (Figure 8). UCM provides the assurance that models and code are up to date.

In a UCM project, a component baseline contains model artifacts as well as code artifacts. Tight integration of the *Rose Model Integrator* with the ClearCase *diff/merge* feature means teams can develop models in parallel as conveniently as they can code. The change set implementation in UCM (see Change Sets) extends to model elements when operating in a UCM project, so that a change set incorporates changes to the code artifacts as well as models. Rational Rose supports the UCM *deliver* and *rebase* operations (see Activity and Artifact Management), enabling multiple designers to work simultaneously on Rose models as effectively as developers can work concurrently on code.

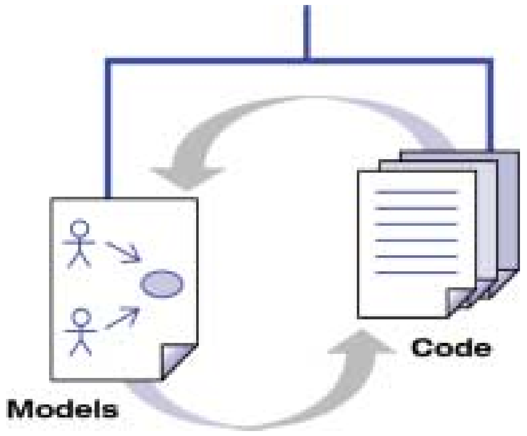


Figure 8: Leading teams move code and model elements through the development process in parallel

Artifacts from Testing

Traditional thinking placed the testing effort at the end of development, ‘after’ the coding effort. Component development severely tests this approach to quality and most projects that approach testing in an after-the-fact manner fail.[3] Leading component development teams know that unit tests are critically important to project success. Rather than leave component quality assurance for integration stages, they proactively test units, subsystems and systems.

All this concurrent testing produces numerous testing artifacts — test requirements, scripts, data and voluminous test results. Rational Suite TestStudio™ and Rational QualityArchitect provide the integrated infrastructure development teams need to consistently produce quality software artifacts. With UCM, teams can manage their testing artifacts concurrently with the associated development artifacts.

Teams rarely require the parallel development of automatically generated testing artifacts, such as Rational Robot test scripts. But most teams realize benefits by integrating these test artifacts with component baselines. In a UCM project, component baselines contain all code artifacts in addition to related test requirements, tests and test data. The change set notion described above extends to test cases, scripts and data used to validate changes. Integrated activity and artifact management links the full scope of an activity — from code changes to validation suites — to one UCM change set described by its activities.

Rational QualityArchitect produces test artifacts that are much closer to code artifacts than what Rational Robot produces (QualityArchitect produces unit tests based on Rose models). With UCM, development teams using QualityArchitect unit testing can manage these test artifacts in lockstep with code development. As the development team writes and modifies components, it simultaneously creates testing infrastructures and unit tests for these standalone components. The UCM component baseline includes both the code and full suite of test artifacts that validate these components. When changes in capacity, assignments or other aspects of the project are made, new developers get immediate access not just to required code artifacts, but also to the validation suites used previously by other team members.

Analysis, Design, Coding and Testing Artifacts

The benefits offered by Rational Suite and UCM are invaluable. Component baselines consolidate all project artifacts in a baseline — from requirements documents to validation suites — a major advantage for any team that needs to perform maintenance or upgrade tasks. With UCM baseline levels, cross-functional teams that use Rational Suite can more easily:

- Identify the required activities for a specific component version
- Determine when to rebase
- Identify activities that require testing
- Assess in-process artifacts

In addition, UCM baseline levels help teams identify when cross-functional activity is appropriate, for example, when a baseline has achieved a quality level appropriate for integration testing. These acceptance levels are often determined by smoke tests that set the bar over which baselines must graduate before integration testing proceeds.

In UCM, quality monitoring is further enhanced with instantaneous reporting functions provided by Rational ClearQuest. With these reporting capabilities, team members can generate clear and concise data on project status to quickly detect problems, assess risk and reach resolution fast. Work is organized around high-priority activities and all releases meet predefined levels of quality.

Conclusion

As the Internet continues to grow as businesses’ primary channel for transacting with customers and gathering strategic information, the pressure on software development teams to create applications that support these functions

[3] “What Are Your Requirements?”, 2000 The Standish Group Research Note, 2000. The Standish Group International.

will increase. This trend escalates the need for predictable, repeatable change management processes in software development organizations of all types and sizes.

With the Unified Change Management capabilities featured in Rational ClearCase, Rational ClearQuest and Rational Suite, Rational Software offers a complete change management solution for controlling change in software development. Based on proven best practices gleaned from years of observation and software development experience, the Rational UCM process provides a practical solution to the challenges of producing high-quality software applications at Internet speeds.

About Rational Software

Rational Software Corporation (Nasdaq: RATL), the e-development company, helps organizations develop and deploy software for e-business, e-infrastructure, and e-devices through a combination of tools, services and software engineering best practices. Rational's e-development solution helps organizations overcome the e-software paradox by accelerating time to market while improving quality. Rational's integrated solution simplifies the process of acquiring, deploying and supporting a comprehensive software development platform, reducing total cost of ownership. International Data Corporation (IDC) has recognized Rational as the leader in multiple segments of the software development life-cycle management market for three years in a row. Founded in 1981, Rational, one of the world's largest Internet software companies, had revenues of $572 million in its fiscal year that ended in March, 2000 and employs more than 3,000 people around the world.

Rational

the e-development company™

Dual Headquarters:
Rational Software
18880 Homestead Road
Cupertino, CA 95014
Tel: (408) 863-9900

Rational Software
20 Maguire Rd
Lexington, MA 02421
Tel: (781) 676-2400

Toll-free: (800) 728-1212
Tel: (408) 863-9900
Fax: (408) 863-4120
E-mail: info@rational.com
Web: www.rational.com
For International Locations: www.rational.com/worldwide