Managing Project Risk

Best Practices and Tool Integration in Software Development

Managing Project Risk

BEST PRACTICES AND TOOL INTEGRATION IN SOFTWARE DEVELOPMENT

TO800; rev. 8/02. Subject to change without notice.

Managing Project Risk

BEST PRACTICES AND TOOL INTEGRATION IN SOFTWARE DEVELOPMENT

Developing software can be a risky business. For experienced project managers, this is nothing new; for some time now, effectively managing risk in software development has been recognized as a key factor in delivering software successfully. But even dramatic advancements in software development techniques, architectures, and platforms proven to be effective in reducing risk for teams of all sizes and tools have not stopped many projects from being cancelled, or delivered late, or from running over budget. The complexity of the applications being developed has outpaced the resources and abilities of many software development teams. As a result, only a small fraction of software development projects are completed on time and within budget. Project managers and team leaders — from seasoned veterans to those spearheading a project for the first time — are focusing closely on reducing risk in software development.

Through years of close work with thousands of development organizations, Rational Software has developed a comprehensive solution that has proven to be effective in reducing risk for development teams of all sizes. This solution is based on best practices of software engineering — including an iterative development process — and a tightly integrated set of development tools that helps teams improve communication and leverage assets throughout the entire project lifecycle.

We've prepared this guide for project managers and team leaders who understand the importance of minimizing risk and want to help their teams realize their full potential. The content demonstrates the value of integrated tools in risk management and guides you step-by-step through a sample project lifecycle. This guide will help you understand how Rational's process, award-winning tools, and key tool integrations can work in concert to help you and your team decrease risk, increase predictability, and ensure that your next project is delivered on schedule.

If you have questions or comments, or if you would like to arrange a demonstration or evaluation of Rational's solutions, please visit us on the Web at www.rational.com.

Contents

Managing Project Risk

BEST PRACTICES AND TOOL INTEGRATION IN SOFTWARE DEVELOPMENT

Iterative Software Development: Managing Project Risk and Uncertainty

Today, software projects are initiated with significant levels of uncertainty:

1. When will code and content be stable?
2. What runtime failures will be uncovered?
3. How do project managers make accurate project progress forecasts to stakeholders and the team?

Because of this uncertainty, it is nearly impossible for project managers to adhere to a predictable project schedule. Yet, successful managers have been able to consistently guide their teams to complete projects on time by adopting proven techniques for effectively dealing with project uncertainty — by following best practices and using integrated tools.

The classic waterfall approach to software development — in which the project proceeds sequentially from requirements analysis through design, coding and testing — postpones addressing risks until late in the project, when it is much more costly to make changes and fix problems introduced in earlier phases. In the search for more predictable projects and risk reduction, many organizations have found the waterfall approach unworkable.

From an overall business perspective, the success of many organizations is becoming increasingly dependent on the success or failure of the software they build — regardless of whether it is intended to be packaged and sold, to be used internally, or to drive business transactions. In this environment, managing risk is not only a sound development practice, but also a vital business practice.

What Is Risk in Software Development?

Software development teams encounter many different kinds of risks. By definition, risk involves an exposure to potential hazards. It is uncertainty that underlies risk. Factoring in this uncertainty and examining the unknown and ill-defined is accomplished by creating and prioritizing a set of risks based on what is known about the project. For example, after an early risk assessment you might conclude:

Risk 1: Requirements evolve over a project. Project scope creep is inevitable

Risk 2: We have no straightforward way of integrating our developments assets. There's lots of manual rework

Risk 3: We know that we will be using XML extensively, but we have only one developer with XML experience

With little effort, you can probably think of a few important risks that your current software development project is facing. Of course, the order of the items on the risk list, as well as the items themselves, will change over time. The important thing is to take steps to address risk and uncertainty as soon as possible, before you spend a lot of time, effort, and money on a flawed approach.

Reducing Risk Through Best Practices

In contrast to the waterfall approach to software development, an iterative approach seeks to identify project risks — both technical and business-related — early in the lifecycle, when it is possible to address them most efficiently. The iterative

development process is driven by continuous discovery, invention, and implementation, during which the project team completes development of project artifacts in a predictable and repeatable way, iteration by iteration. An iterative process enables your team to mitigate risks earlier because it unearths and tests for problems earlier. As you progress through the initial iterations, you exercise many aspects of the project, including tools, off-the-shelf software, your process, and the skill sets of individuals on your team.

The main advantage of iterative software development is that it increases the predictability of the schedule, the final product and the entire process. As an additional dividend, it can also be expected to produce higher quality products — that satisfy the real needs of end-users — because the requirements can evolve along with the design and the implementation. In fact, developing software iteratively is one of six best practices of software engineering that are commonly used throughout the industry by successful organizations. This time-tested approach to software development is embodied in the Rational Unified Process®, Rational's comprehensive framework for software development. It is a cornerstone of Rational's complete solution, which combines best practices, unified tools, and services that accelerate implementation.

Rational Software Best Practices

- *Develop Iteratively* – to identify and eliminate risks before they threaten your project.
- *Manage Requirements* – to ensure resilience in the face of inevitable change.
- *Use Component Architectures* – to make your architecture tangible to all practitioners.
- *Model Visually* – to attain and preserve a high-quality architecture.
- *Continuously Verify Quality* – to ensure quality throughout the development life cycle.
- *Manage Change* – to enable efficient parallel development within teams and across the enterprise.

Integrated Tools Pave the Way

Adopting an iterative development approach does not necessarily mean the project manager will have an easier job. In some situations, iterative development requires more careful planning and will likely place an additional burden on project managers. But the benefits of iterative development far outweigh the costs, especially when the development team is equipped with a set of tightly integrated, team-based tools that support the entire lifecycle. With Rational's integrated tools, software development artifacts, such as requirements specifications, architectural and design models, and test cases, can be leveraged and reused repeatedly, and all team members have clearer, more reliable mechanisms for conveying and understanding project details. Integrated tools help teams of analysts, developers, and testers work closely together, by providing an infrastructure that allows each team member access to common project requirements, defects, test cases, and more. In addition, tool integration increases the return on investment by enabling round-trip engineering and synchronization of requirements, design elements, and test artifacts. In short, Rational's broad array of integrated tools work together to minimize project risk and ensure successful project delivery in several vitally important areas:

- Increased productivity and efficiency
- Faster development cycles and improved ability to meet deadlines
- Improved team communication
- Increased quality
- Simplified project management

Rational Product Integration and Workflow

Full Lifecycle Integration

Integrations among tools offer considerable advantages that have a cumulative effect. When a team uses two or three integrated tools, there is typically a noticeable boost to team productivity and cohesiveness; but when that same team can rely on a comprehensive set of integrated tools that spans the entire software development lifecycle, the benefits — including improved quality, better predictability, and faster development cycles — have a substantial positive impact on the entire organization. Rational tools offer such benefits.

For the extended development team — including project managers, analysts, developers, and testers — to work closely together, they need a solid infrastructure. As a project manager, you must ensure that all team members have access to the requirements, defects, test status, and other key project artifacts. Automation and integration throughout the project lifecycle help unify software teams, improve organizational proficiency, and capability and keep projects on schedule.

Project Workflow

The value of individual integration points between Rational tools is perhaps best understood in the context of the development process and workflow. In its simplest form a process describes who is doing what, how, and when. In the Rational Unified Process, workers represent the "who" activities represent the "how" artifacts represent the "what", and workflows define the "when". A workflow is simply a sequence of activities that produces a result of observable value.

In the workflow shown in Figure 2.1, the blocks represent artifacts. Those in the top row — Business Model, Requirements and Use Cases, Model, Code, and Unit Tests — are development artifacts. Blocks in the bottom row — Test Plan, Test Cases, System Tests, Test Results, and Defects – are quality assurance or testing artifacts. Above each block are the names of Rational tools that are used to build, maintain, manage, or manipulate the associated artifact. Arrows between blocks depict activities that are supported and often automated by Rational tools. Note that this entire workflow is based upon a solid foundation of Process, Metrics, and Software Configuration Management. You may find it useful to refer to this diagram throughout this section and again in later sections, as elements of this workflow are discussed in greater detail.

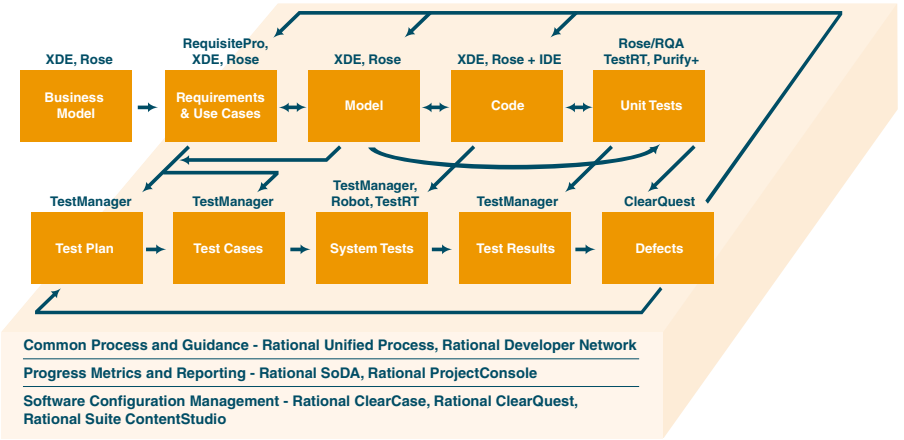

Figure 2.1: Rational Product Integration and Workflow. Connecting lines illustrate where Rational tools automate workflow throughout the software development lifecycle.

Leveraging Assets Throughout the Process

Every software development process produces artifacts as a project progresses from its earliest stages to completion. These artifacts — including requirements, models, test cases, defects and enhancement requests — fall into distinct categories, but they're all inherently related. Through effective use of Rational's integrated tools, you can truly capitalize on the essential relationships among these artifacts. When you leverage work completed in one phase repeatedly in later phases, your artifacts become more than just a byproduct of development — they become valuable assets that you can reuse again and again.

Through integration, your team has easy access to the up-to-date artifacts relevant to their work. The information these artifacts represent is constantly changing, and the most effective way for the team to stay connected is to ensure that everyone is operating with the same current information, building a system based on a common understanding and agreement. Consider how each of the artifacts shown in Figure 2.1 is leveraged through tool integration to drive the entire process. Listed below are common activities in software development and a brief description of the Rational tools that solve project management challenges related to each activity. Integration points related to these activities are presented later in the guide.

Business Model – Define and Solve the Right Problem. When a team models its business process in Rational Rose®, the world's leading tool for model-driven development, software and business professionals communicate more effectively. The output from business engineering is used as input to the software development effort. Rational Rose uses a common process and a common language — the Unified Modeling Language or UML — for both communities, assuring a shared understanding among all stakeholders of what business process needs to be supported in the organization.

Requirements and Use Cases – Unify Developers, Designers, and Analysts. Effective requirements management is common to virtually every successful software project. Rational® RequisitePro® combines an intuitive interface that works with Microsoft Word with powerful search, prioritization, and management capabilities to help teams establish and maintain requirements as they evolve. Integrated Use-Case Management (IUCM) links requirements in Rational RequisitePro with use-case model elements in Rational Rose, providing developers and designers with instant access to the current use-case specification, attributes, and traceability.

Model and Code – Use Round-Trip Engineering to Promote Re-Use, Save Time. Rational Rose helps teams communicate clearly, manage complexity, define higher-quality software architectures, increase component reuse, and capture vital business processes. With round-trip engineering in Rational Rose, your developers can automatically generate source code from application models, and update or create models by reverse-engineering code.

Unit Tests – Find Defects Early. With Rational Rose and Rational® QualityArchitect, your team can automatically generate unit tests — including a test driver and stub — code directly from the Rational Rose model, enabling unit- and scenario-based testing before all components are coded. In addition, developers can use Rational® PurifyPlus to run unit tests on their code, pinpoint hard-to-find bugs, highlight performance bottlenecks, and identify untested code. Results from these tests can be used to automatically generate

defect records in Rational® ClearQuest®, Rational's change management tool.

Test Plans and Test Cases – Ensure Requirements are Tested. Test plans and individual test cases are closely related to project requirements. Managed in Rational® TestManager, the test plan and test cases are linked to requirements in Rational RequisitePro and model elements in Rational Rose. This integration helps ensure that all requirements are fully tested before release. Additionally, when a requirement changes, the associated test case is automatically flagged as "suspect," highlighting tests that may need to be updated.

System Tests – Ensure Reliable Performance and Test System Functionality. Thorough functional tests and complex multi-user performance tests are crucial to product quality and user satisfaction. Integration between Rational® TestManager and RobotJ allows quality engineers and other team members to launch tests and monitor test plan progress with a single, extensible test management tool. In addition, the strong interrelationships among UML models, code, and test cases have been leveraged by Rational® Test RealTime to accelerate the entire development and testing process for embedded software.

Test Results and Defects – Automatically Generate Defects From Test Results. Results from unit tests and system tests can be easily accessed by QA or QE managers, business analysts, developers, and testers. Rational TestManager helps project managers and the entire team track project progress against the test plan throughout development. Defects or other problems discovered during testing can be automatically entered into Rational ClearQuest, where they are easily captured, tracked, and managed along with enhancement requests and other change requests. In an iterative process the story does not end there. The integration between Rational RequisitePro and Rational ClearQuest allows users to associate requirements with related enhancement requests or defects, so that requirements can be added or updated as necessary.

A Foundation of Process, Metrics and Software Configuration Management.

The artifacts, activities, and workflow shown in Figure 2.1 all rely on a solid foundation of process, metrics, and software configuration management. The process provides guidance and direction for individuals and the team as a whole. It also specifies when and what artifacts are to be developed, and it offers criteria for monitoring and measuring progress. Without dependable means of measuring progress, a project manager has no feedback by which to assess the project status and make appropriate adjustments. And those changes, along with changes to requirements, models, code, documentation, and other artifacts must be managed effectively to ensure the efficiency and repeatability of the process.

Common Process and Guidance – Right-size Your Process With the Right Amount of Guidance. The Rational Unified Process (RUP®) is a Web-enabled knowledge base of software engineering processes and best practices that provide your team with guidance to streamline development activities. Your team can access context-sensitive RUP guidance from within all Rational products. For additional direction and expertise, your team can access the RUP Exchange and the RUP Knowledge Center on the Rational Developer Network℠. RUP is designed to help your team "right-size" a process that fits your needs. Platform, tool, and team size Plug-Ins are available for .NET, J2EE, small teams and more.

Progress Metrics and Reporting – Get Instant, Accurate Understanding of Project State. Project managers must have a clear picture of where their projects stand at all times. Rational® ProjectConsole automates the process of researching and reporting project status. Through integrations with the Rational development platform, it dynamically creates a project Web site with a progress metrics dashboard, based on your up-to-date project data. For more traditional project documentation, Rational® SoDA® extracts information from multiple data sources — Rational TestManager, Rational® ClearCase®, Rational ClearQuest, Rational RequisitePro, and Rational Rose -- and automatically generates customized system documentation and reports.

Software Configuration Management – Manage Activities and Artifacts Throughout the Lifecycle. Rational ClearCase, when combined with Rational ClearQuest, is the market leading software configuration management solution for managing change and complexity in software development. Tight integration between Rational ClearCase and Rational ClearQuest allows individual defects and enhancement requests to be associated with the specific version or release of software in which they were fixed or implemented. Rational ClearCase is integrated with a full range of development tools and IDEs. It enables your team to baseline requirements and tests assets, and provides developers with convenient version control of models and source code.

Figure 2.2: The Rational Suite Product Family. Provides integrated solutions for the entire team.

Rational Suite Product Family: Comprehensive Solutions

The Rational Suite® product family (figure 2.2) offers comprehensive, integrated solutions that unify development teams and simplify the job of developing software. Each Rational Suite provides solutions tailored for different practitioners. The Rational Suite Product Family includes:

- Rational Suite® AnalystStudio®
  - Rational Rose Professional Data Modeler Edition
  - Rational RequisitePro
  - Rational ClearQuest
  - Rational ClearCase LT, and the other Team Unifying Platform tools
- Rational Suite DevelopmentStudio
  - Rational Rose Enterprise Edition
  - Rational QualityArchitect
  - Rational PurifyPlus and the other Team Unifying Platform tools
- Rational Suite® TestStudio®
  - Rational Robot J
  - Rational® TestFactory®
  - Rational PurifyPlus
  - Rational TestManager and the other Team Unifying Platform tools
- Rational Suite Enterprise — Rational's most comprehensive solution features all the tools of Rational Suite AnalystStudio, Rational Suite DevelopmentStudio, and Rational Suite TestStudio

The Team Unifying Platform

Rational's Team Unifying Platform is a unique set of team-based tools and resources that provide a foundation for effective communication, collaboration, and testing. The Team Unifying Platform is an integral part of each Rational Suite product and plays a key role in helping groups of individuals work together as a team. The Team Unifying Platform features tools for collaborative project management:

- Rational ProjectConsole
- Rational Unified Process
- Rational SoDA

and for team infrastructure:

- Rational RequisitePro
- Rational TestManager
- Rational ClearQuest
- Rational ClearCase LT

Rational ProjectConsole: Immediate and Accurate Project Metrics For The Entire Team

Without a clear picture of where you are, where you've been, and where you're going, your ability to guide projects to successful completion is severely impaired. Knowing this, it can still be quite a challenge to gather accurate metrics and report on project status. How do you really know the true state of your project? Status meetings, e-mails, hallway conversation — even weekly reports — are mostly based on subjective rather than objective analysis.

Rational ProjectConsole automates the process of gathering project metrics and dynamically creates an objective project Web site. The Web site features a progress metrics dashboard based on data collected from Rational Suite and other third-party data sources. (Figure 2.3) ProjectConsole automatically builds, updates, and maintains the team Web site providing one centralized place for project team members to access all project artifacts and status information. Project managers and team members no longer have to manually gather status updates. In summary, ProjectConsole:

- Provides project teams with up-to-date access to comprehensive project information
- Automatically collects and publishes information — on demand or on a scheduled basis
- Quantitatively and objectively measures project progress and quality

Rational ProjectConsole collects standard and custom metrics from Rational Suite tools — such as Rational Rose, Rational RequisitePro, Rational ClearCase, Rational ClearQuest, and Rational TestManager. It also supports third-party products, including Microsoft Project. Hundreds of sample reports and templates are available out-of-the-box.

Through charts and indicators, team leaders can quickly evaluate the progress and quality of their project. If certain aspects of the project are lagging behind schedule, they can take corrective action, by allocating more resources, for example.

Rational ProjectConsole is much more than a high-level tool for team leaders. It provides all members of the development team with the ability to drill down and analyze the low-level details, planned-versus-actual metrics, and historic data. These capabilities enable the software development team to discover the root causes for late deliverables, set realistic project expectations, forecast future project milestones, and objectively and accurately measure project progress.

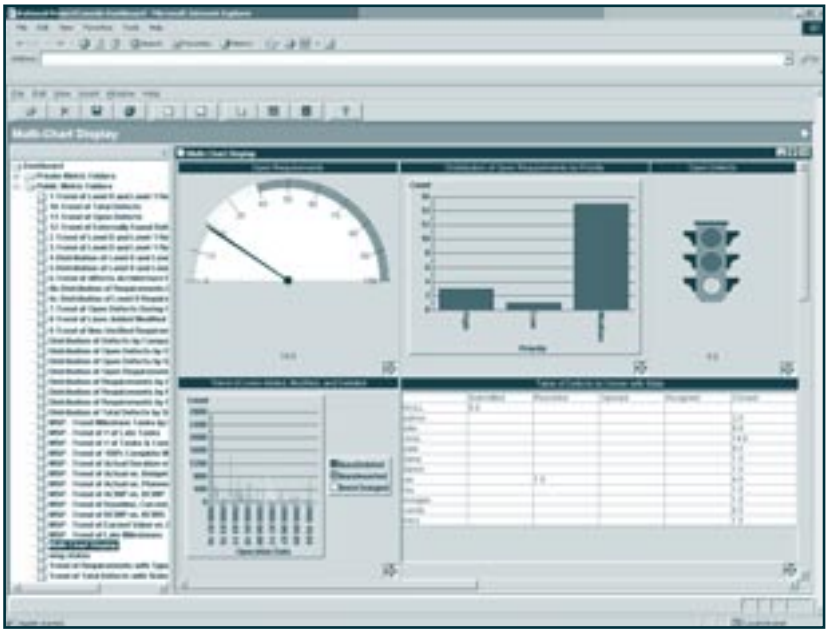

Figure 2.3: Rational ProjectConsole. Provides project leaders and other team members with analytical tools to assess project status as well as access to detailed metrics and project artifacts.

The Rational Approach: Integrated Tools and Best Practices

The following sections will focus on four major disciplines of software development: Requirements Management, Visual Modeling, Software Configuration Management (SCM) and Automated Testing. These areas are closely linked to the best practices of software development, and all of them have inherent challenges.

A Real World Example: How Do Rational Tools Integrate?

To illustrate how Rational tool integrations combined with a best practices approach to software development helps minimize project risk in real-world applications, each of the following sections traces a fictional — but not unrealistic — project from enhancement request, through to requirements, modeling, SCM, testing, and project readiness. Along the way, you will learn how Rational tools and key integrations are used to meet the challenges of each individual discipline, and how the tools help meet the unique challenges faced by project managers.

The example project centers on a fictitious online retailer, ClassicsCD.com. ClassicsCD.com sells classical music to its customers through its Web site. The example project starts with a customer request. This request becomes a project requirement that will be implemented as a new feature and tested. Note that this example cannot possibly illustrate all of the benefits and potential applications of Rational's tools. Each organization, and each project, has a unique set of needs and constraints. Some may be just starting a new development effort, while others, like ClassicsCD.com, are working on the next release of existing software. As you follow the ClassicsCD.com project, you'll be able to see parallels with your current or previous projects, as well as opportunities for increasing the productivity of your team and the predictability of your development efforts.

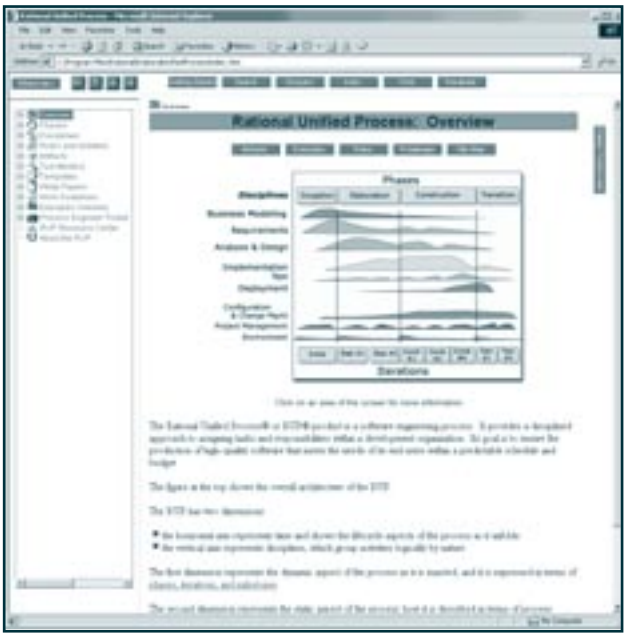

Figure 3.1: The Rational Unified Process. Shows teams how they can apply best practices of software engineering, and how they can use tools to automate the software engineering process.

Example Project: Product Marketing Receives Customer Enhancement Request and Submits To Project Lead

While placing an order, a customer notices that the order confirmation page does not say when ClassicsCD.com expects to ship the order. This customer decides to fill out a customer feedback form, and requests that the confirmation page include the ship date of the order.

Feedback is directed to product marketing managers. In this case, the product marketing manager recognizes that the suggestion is worthwhile and enters an enhancement request in the project's change request database, using Rational ClearQuest Web.

The product marketing manager creates a new enhancement request, enters a description of the request, sets the priority of the request as "High", and records the customer's name and e-mail address. With Rational ClearQuest, everyone in the organization submits change requests in a consistent, easy-to-manage way, so nothing gets lost.

Project Leader Assesses Workload and Assigns Tasks

The project leader for ClassicsCD.com relies on Rational ClearQuest to help him manage change throughout the entire development process. He has configured Rational ClearQuest to run a query automatically each time he logs in. This query finds all unassigned enhancement requests. On this day, the query finds the new Ship Date request. After reviewing it, he decides to assign it to a team member for immediate action. Using Rational ClearQuest, he:

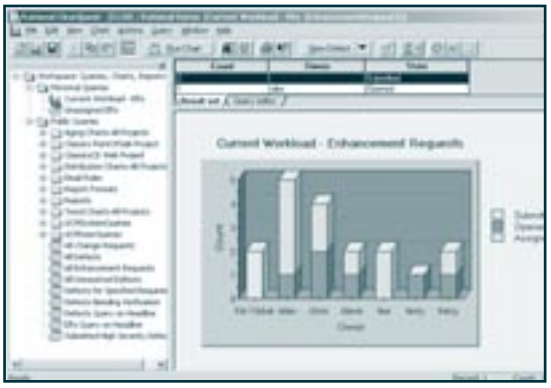

Figure 3.2: Rational ClearQuest. Helps identify overloaded and available team members.

- Runs a query to determine the current workload of his team members (Figure 3.1)
- Identifies a project system analyst or developer with a relatively light workload
- Assigns ownership of the enhancement request to her, adding any other notes or comments he thinks would be helpful

Already, a customer's idea has been captured, documented, and assigned to an analyst or developer in a manageable, reliable, and consistent way.

Process and Tool Help — Anytime, Anywhere

At any time, anyone on the team can refer to the Rational Unified Process (Figure 3.1) for guidance on methodology and specifics on tool use. With RUP the entire team can:

- Review process information on workflows, roles and activities
- Access technical detail on how to do the work
- Quickly view context-sensitive help for Rational tools

This allows people of different experience levels and abilities to get just what they need from the process when they need it. In this sense, RUP is not only a process, but also a knowledge base, delivering the benefits of Rational's extensive software development experience to your team.

In addition, ClassicsCD.com team members can turn to the Rational Developer Network for topical discussions, white papers, online training, and artifacts tailored exclusively for the worldwide community of Rational development professionals. The Rational Developer Network can be an invaluable resource for project leaders who want to keep up to date with the latest industry trends, techniques, and tools.

Requirements Management

For project leaders, defining and analyzing requirements are crucial and challenging aspects of the project. Project leaders must simultaneously address multiple demands, including the need to manage the enormous complexity of projects, the need for greater predictability throughout each project, and the need to manage skills and resources. The ever-increasing complexity of software development projects can make it difficult to perform fundamental project management tasks, such as assessing the completeness of requirements, reviewing priorities and project status, balancing new functionality with maintenance, and developing an accurate schedule based upon requirements. To ensure the predictability of a project, a project leader must understand the time and cost implication of changing requirements, and be able to analyze the impact of change on the overall project timeline. Lastly, an iterative development approach, while offering extensive advantages in reducing project risk, requires effective project management to ensure that everyone on the project team is working on the correct priorities.

Numerous studies conducted on software development projects have repeatedly found that the most common causes of project failure are related to requirements. Clearly, the better you communicate and manage your requirements, the better chance you have of guiding your projects to successful completion.

Managing Requirements with Rational RequisitePro

Effective requirements management starts with Rational RequisitePro. Rational RequisitePro bridges the gap between technical and non-technical users, facilitating better communication while helping teams understand the impact of changing requirements. With Rational RequisitePro you can create, review and edit requirements using a familiar Microsoft Word interface. Rational RequisitePro directly links the requirements in your document to a database where they are easily accessed, organized, and traced by your development team (Figure 4.1). In Rational RequisitePro you can link related requirements together. When changes to one requirement affect others, project managers are automatically alerted and can instantly pinpoint and assess the change's impact. With this ability to perform impact analysis, you can make quick, informed decisions for scope management or resource reallocation. Rational RequisitePro also includes a Web interface, providing all team members easy access to requirements, even in remote locations or multi-platform environments.

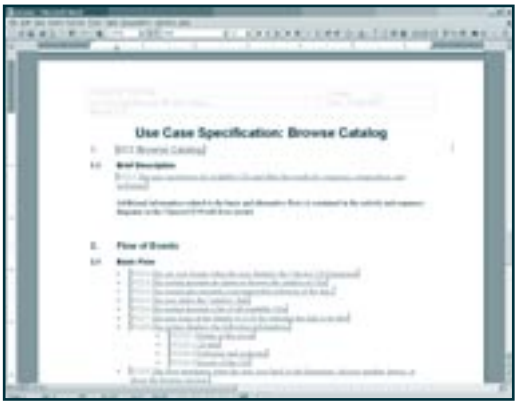

Figure 4.1: Rational RequisitePro. Dynamically links requirements in Word documents with a database allowing all team members to access requirements in the format most appropriate for them.

Integrating Rational RequisitePro with other Rational Tools

A key part of effective requirements management is problem analysis — gaining a clear understanding of why a project is being built. To create a system that really solves the customer's needs, you must ensure that needs from all stakeholders are appropriately taken into account, using a combination of problem analysis, requirements management, and business modeling — with Rational Rose for example.

As project requirements take shape, analysts begin to transform them into a design for the new or enhanced system. From use-case specifications and other requirements, an architecture for the system evolves. The tight integration between Rational RequisitePro and Rational Rose streamlines this process. Through its integration with Rational Rose, Rational RequisitePro connects requirements with use-case models, giving developers instantaneous access to the use-case specifications from use-case diagrams, as well as visibility into all requirements information. This integrated use-case management helps ensure your team will implement a system that accurately meets customer needs.

Project managers need to track stakeholder input on an ongoing basis. Integration between Rational RequisitePro and Rational ClearQuest enables you to associate requirements to related enhancement requests or defects. Tracking these changes ensures that all qualified stakeholder input is communicated, addressed, and tracked from requirements to release.

To further simplify change management, the integration between Rational RequisitePro and Rational ClearCase, Rational's software configuration management solution, enables you to include requirement assets as part of a project baseline, and to create new Rational RequisitePro projects from existing project baselines. Of course, changes to use case models and other models in Rational Rose can be comprehensively managed with Rational ClearCase as well. Unified Change Management (UCM), which will be discussed in more detail later, allows you to associate specific changes to models, source code, and other artifacts with the particular change request or defect that the changes address.

Testers who play a crucial role in reducing project risk, must know exactly what the project requirements are at all times to ensure their test coverage is as complete as possible. The integration of Rational RequisitePro and Rational TestManager allows requirements to serve as direct inputs to test case creation, so that testers and QA engineers can validate the system. And, as requirements change, you and your QA team can instantly see what test cases are impacted.

Example Project: System Analyst Automatically Receives Tasks And Is Backed-By Automatic Change Tracking

Creating a Requirement from an Enhancement Request

Once the project manager delegates an enhancement request, it is up to the project system analyst to create a formal project requirement from it. When she logs into Rational ClearQuest, she sees that the new request has been added to her "To Do" list, which is updated automatically. She notices the request is marked as high priority and decides to start right away.

Realizing that the ClassicsCD.com use-case model will need to be modified, she uses Rational Rose to:

- Check out the sub-model packages that will need to be modified
- Specify the Ship Date enhancement request as her current activity

With the model under ClearCase/Unified Change Management (UCM) control, this straightforward process automatically links the changes she is about to make with the request she is working on, so that they can be tracked together (Figure 4.2).

Continuing, she:

- Opens the main use-case diagram for the project and identifies the "Arrange Shipment" use case
- Views the requirements associated with that use case by leveraging the power of Integrated Use Case Management – and simply right-clicks on the use-case diagram to open the Word document, which is automatically positioned to the appropriate location to enter the change

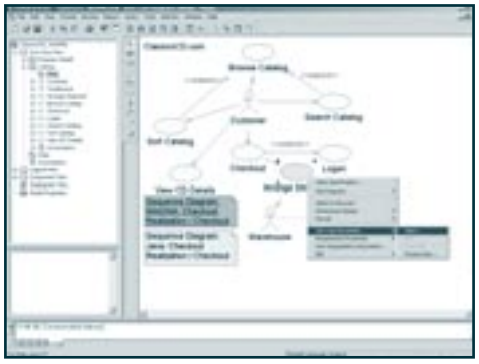

Figure 4.2: Integrated Use Case Management. Use case models in Rational Rose are tightly coupled with requirements in Rational RequisitePro.

Now in Rational RequisitePro, the project system analyst:

- Types "The warehouse system responds with the estimated shipping date for the product(s) in the user's shopping cart," into the Word document
- Selects the text and right-clicks to make it into a new, formal project requirement
- Optionally associates the requirement with other related requirements and adds information as needed
- Saves the document

At this point the new requirement is created in the Rational RequisitePro database, and the customer's request becomes an "official" project requirement visible to everyone on the team. Developers can see a new requirement that must be coded, and testers see the same requirement, which they must validate.

Updating the Data Model

To support this functionality the database design will need to change. With Rational Rose unified modeling capabilities, your business, application and data modelers are all connected. Using Rational Rose, the system analyst:

- Adds "Shipping Date" to the ORDER object in the database
- Transforms the Object Model into a Data Model, and transforms the logical data types into the correct physical data types for the application's database
- Checks the visual model back in

Because ClassicsCD.com is using the UCM integration between Rational ClearQuest and Rational ClearCase, the visual model is checked back in, when UCM remembers the activity and lists all of the changed files in the change set for that activity. As a final step, the system analyst associates the original enhancement request with the new use case requirement that will fulfill it. She modifies the enhancement request record in Rational ClearQuest and adds the new use-case requirement to its requirements list.

Summary of Key Rational RequisitePro Integrations

- Integration with Rational ClearQuest enables you to associate requirements to related enhancement requests or defects, ensuring that all stakeholder input is accurately communicated and addressed.
- Integration with Rational ClearCase provides an efficient mechanism for requirements reuse and archiving. Using Rational ClearCase and Rational RequisitePro, you can capture a set of requirements at each iteration of the project, or at other periodic intervals.
- Integration with Rational Rose – known as Integrated Use Case Management – helps ensure that you are solving the right problem and developing the right solution. Requirements in Rational RequisitePro are connected to Rose use-case model elements, providing analysts, developers, and designers with instant access to the use-case specification, attributes, and traceability.
- Integration with Rational TestManager ensures that your testing team is always validating against the most up-to-date requirements. When requirements are used as test inputs for test cases, your team can develop and maintain a test plan in line with project requirements, and generate comprehensive test coverage reports. Plus, with this integration, suspect test cases are automatically highlighted when related requirements change.

Visual Modeling

For decades civil, mechanical and electrical engineers have been creating models of what they build, from bridges and buildings to airplanes and microprocessors. The designs they model are broken down into components, such as struts, beams, turbines, and memory units. The reasons for modeling and using component architectures are compelling. The economic cost of constructing the actual projects is enormous, as is the cost of failure should a bridge collapse or an airplane break down. Modeling provides engineers with the ability to assess different architectures, to visualize entire systems and to communicate designs more clearly. Likewise, using component architectures facilitates modular designs, resilient architectures, and parallel development.

Software systems have become so complex, and the cost of failure so high, that modeling is no longer considered a luxury or an option, but rather a necessity for all but the most trivial projects. Successful development organizations understand that application quality starts with good design. Visual modeling and component architectures eliminate ambiguity; use cases precisely define behavior, and models accurately capture design and reveal inconsistencies. Clear communication, better design, and less uncertainty all add up to a less chaotic, more predictable development effort.

Modeling with Rational Rose

Both industry experts and users alike have recognized Rational Rose as the world's leading visual modeling solution. In Rational Rose, all models — including business models, application models, and data models — are managed and maintained in a single tool, using a single language: the Unified Modeling Language™ or UML (Figure 5.1). Rational pioneered the UML, the industry standard notation for specifying, visualizing, and constructing software and systems. Rational Rose features powerful forward- and reverse-engineering capabilities for numerous programming languages and databases. Your team can make changes in the model, the code, or both, and easily re-synchronize them at any time. Through integration of the modeling and development environments, Rational Rose simplifies many development tasks and increases individual productivity. In addition, with your design accurately documented in visual models, the learning curve for new team members is dramatically shortened.

The earlier you can begin verifying quality in your development process, the better your chances of finding and fixing problems when they are least difficult and costly to repair. Using Rational Rose to visually model your software is, by itself, an exceptionally effective way to reduce risk by ensuring that your software's architecture and design are sound. In addition, Rational QualityArchitect, an extension of Rational Rose, helps teams unit test individual com-

ponents before the entire system is assembled, by automatically generating test drivers and stubs directly from your models in Rational Rose. And, the Rational XDE product family provides tightly integrated support for Microsoft® Visual Studio™ .NET as well as full integration with the IBM WebSphere Studio Application Developer and Eclipse Integrated Development Environments (IDEs). Your developers can work in a single environment from design to code and use XDE's user-definable patterns and code templates to automate repetitive coding tasks.

Example Project: Checking-Out Artifacts and Adding a New Operation to the Code

Once the project system analyst has made the necessary modifications to the use case model and the data model in Rational Rose, it is time for one of the developers to update the application model and make the necessary code changes to add the Ship Date to the order confirmation screen. Just as the project system analyst did, the developer will use Rational ClearCase and UCM to check out the necessary models and files and associate them with the activity he is working on. Rational Rose extends the Unified Modeling Language to enable developers to easily represent elements of a Web site, things like client HTML pages, Active Server Pages, Java Servlets, and forms. In Rational Rose, the developer:

- Reviews a diagram of the Web site's home page
- Locates the Cashier page, which references the visitor's shopping cart and looks up Customer, Order, and Order Item information in the database
- Adds a new operation to this object class to implement the enhancement request — show when ClassicsCD.com expects to ship each order

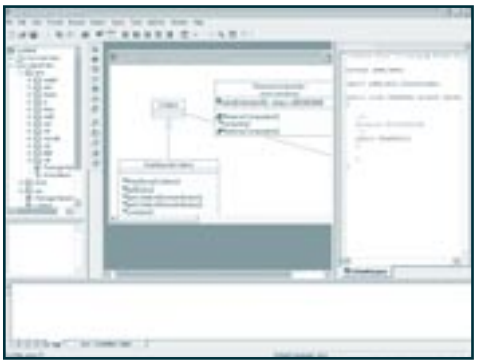

Figure 5.1: Rational Rose. With Rational Rose and the UML, your team can create and maintain clear, unambiguous visual models of your applications and communicate the design accurately to everyone on the team.

Sequence diagrams are graphical views of a scenario that show object interactions in a time-based sequence. While still in Rational Rose, the developer:

- Displays a sequence diagram in which the class participates to see how it is used in the application
- Adds a new operation, "getShipDate" to the sequence diagram
- Generates source code for the new operation in Java, Visual Basic, or another language

Maintaining Synchronization between the Model and Source Code

As developers write code, they may find that additional changes are required; for example, a function may need an additional argument passed to it. At this point, the developer could go back to the model, add the parameter, and re-generate the code. Or, he could add the argument to the function in the source code and then update the Rose model automatically, using the reverse-engineering capabilities of Rational Rose. The ability to perform round-trip engineering helps teams keep project models and the underlying source code synchronized at all times.

Once he has finished coding and updating the model, the developer checks in his work, and again, UCM automatically links the changes he's made to the "Display Shipping Date" activity that he is addressing. After any change, the project manager can review what activity was addressed and be confident that the model still accurately represents the underlying source code.

Summary of Key Rational Rose Integrations

- Integration with Rational RequisitePro enhances Rational Rose use-case modeling with powerful requirements management capabilities. By extending use cases beyond diagrams with sortable attributes, documents, and traceability, this Integrated Use-Case Management helps you manage large numbers of use cases as they are defined, implemented, and tested across your team.
- Integration with Rational ClearCase provides a managed, controlled environment for all model changes as well as powerful merge capabilities to simplify parallel development. Rational Rose shows which models are checked out, while UCM integration tracks what model changes were made to complete each development activity.
- Integration with Rational TestManager enables you to associate test inputs for test cases with model elements from a Rational Rose model. These associations can be used to analyze model test coverage and to validate process and flow represented within a Rational Rose sequence diagram.

section 6

Software Configuration Management

Perhaps more than anyone else in a software development organization, project managers are keenly aware that everything in software development is constantly changing. Requirements, models, source code, tests, as well as processes, technology and personnel — all of this is in motion at once. Managed change is good; it reflects a willingness and ability to adapt and deliver useful and valuable software solutions. But unmanaged change is simply chaos — unproductive and frustrating for everyone involved. Effective software configuration management is especially important in an iterative development process, during which many artifacts are frequently modified.

Software configuration management (SCM) is a combination of two distinct disciplines: software artifact management and defect and change tracking. Software artifact management is concerned with what changes were made, at what time, and by whom. Defect and change tracking identifies why a change was needed, who requested it, how urgent it is, and how close it is to being completed. While both disciplines can be undertaken independently, there are significant benefits to linking them together and tackling them with integrated tools.

For project managers, SCM is about "abilities": scalability, repeatability, traceability, and ultimately predictability.

- **Scalability** – Today your project team may be just a small group working on 4,000 lines of source code. Next year you may have hundreds of people working in multiple, distributed locations on 4,000,000 lines of source. In either case, you need to manage change well, and preferably without having to learn new tools and processes as your projects and teams grow.
- **Repeatability** – Effective software artifact management for all development assets, including the ability to roll back to earlier versions and merge parallel development streams, is essential to establish a repeatable, reliable and efficient development process.
- **Traceability** – To avoid unnecessary rework and ensure that all necessary tasks are completed, project managers need to know who made any given change, why they made it, and when. In addition, they must be able to get both high-level and detailed views of project status, progress metrics, developer workload and release feature sets.
- **Predictability** – When you control change successfully, you can see where your project stands without guessing. You can make decisions based on fact, not assumptions. And you can accurately predict what functionality will be in what release and on what date.

Managing Change With Rational ClearCase and Rational ClearQuest

Rational ClearCase, Rational's award-winning software configuration management (SCM) tool, combined with Rational ClearQuest, the most flexible defect and change tracking tool on the market, creates a solution that is scalable, repeatable, traceable, and predictable. Rational's SCM solution helps you manage even the most complex change throughout the development lifecycle, while freeing your team from tedious tasks that sap productivity.

Rational ClearCase and Rational ClearQuest are seamlessly integrated, and can be used in combination to deliver Unified Change Management (UCM), a

powerful out-of-the-box workflow for automating change (Figure 6.1). UCM expands upon the features and benefits of that integration, giving development teams a workflow that defines and organizes work around activities and artifacts within the project. An activity is a unit of work, such as an enhancement request or defect, that an individual can perform. Artifacts are items such as source code, models, or Web pages. UCM integrates the activities managed through Rational ClearQuest with the artifacts managed through Rational ClearCase, simplifying the process of change. In UCM, activities, such as defects or enhancement requests, are linked with the artifact modifications that were required to implement them. For example, while viewing a list of recently completed activities in Rational ClearCase, a developer can easily view the Rational ClearQuest enhancement request or defect record for each one. From there he can drill down and access the activity's change set, and even perform additional Rational ClearCase operations on files in the change set.

Rational ClearCase, Rational ClearQuest and UCM help project managers reduce risk by coordinating and prioritizing the activities of developers and ensuring that they are always working with the right sets of artifacts. Extending across the lifecycle to accommodate all project domain information — requirements, visual models, code, and test artifacts — they allow teams to baseline requirements together with code and test assets.

In addition, Rational ClearCase provides straightforward parallel development to accelerate release cycles. And Rational ClearQuest can be customized to provide team members with the timely, relevant project information they need to get their job done with maximum efficiency. Your team gets powerful, easy-to-use tools to manage their workload and eliminate tedious tasks. You get measurable progress and visibility into problem areas, trends, and resource allocation, making it easier to complete more projects within budget. And, most important, your users or customers get higher quality software on time.

Example Project: The Developer Tests and Delivers

Throughout the day, he used Unified Change Management to manage his activities. At this point, developers in some organizations perform unit tests on their own work, either manually or by using automated profiling tools such as Rational PurifyPlus for runtime error detection, performance bottleneck identification, and code coverage analysis. Once the developer has completed the required changes,

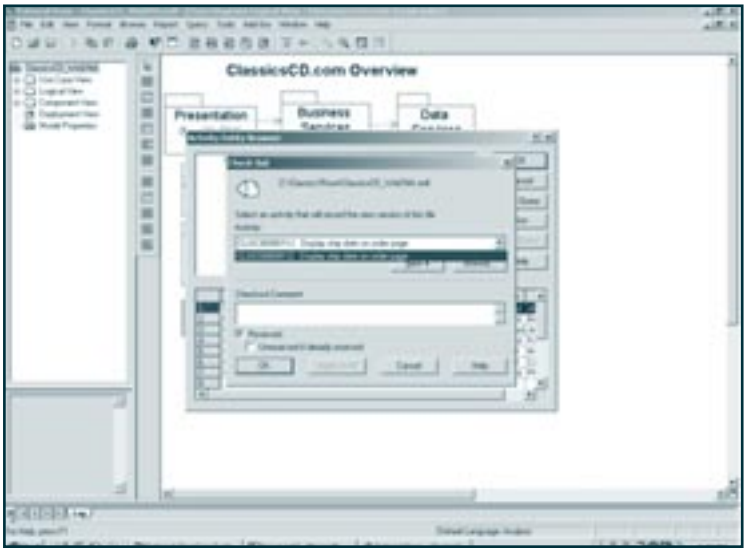

Figure 6.1: Rational ClearCase and Rational ClearQuest. Both applications are fully integrated and enable users to switch seamlessly between them.

he is ready to deliver the results of his work from his personal workspace. In Rational ClearCase Explorer, the developer:

- Views a list of all of the activities he worked on.
- Optionally reviews the change set for any activity listed, and sees a list of every version of every file that he changed while completing this activity. In this case, these include visual model files and source code files.
- Delivers the activity — not just files — from his personal development workspace to the project's integration stream. He can select specific activities whose change sets he wants to deliver, or he can accept the default and deliver all of the activities that he worked on. By managing changes at the activity level, UCM abstractions makes even the most complex merge operations easier.

Streamlining Parallel Development

Rational ClearCase and UCM offer unsurpassed support for parallel development. If anyone else on the ClassicsCD.com team changed some of the same files, UCM simplifies the process of merging the changes from all developers (Figure 6.2).

Figure 6.2: Rational ClearCase. Simplifies the process of resolving conflicts while merging source code, or even XML, as shown here.

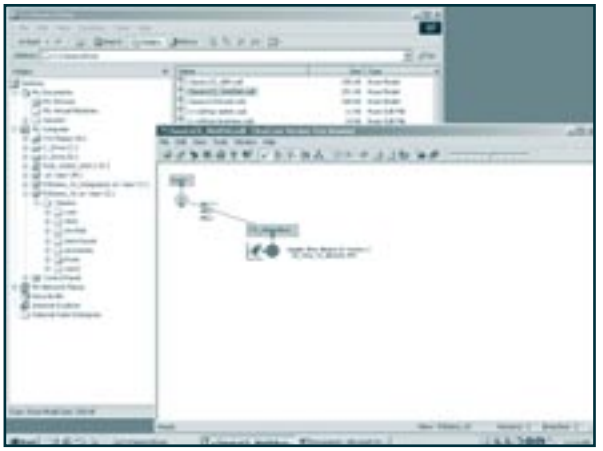

Figure 6.3: The Rational ClearCase Version Tree. Provides a graphical history of an individual element.

UCM handles trivial changes automatically. For example, if two different functions were modified by different developers, then the merge would be completed without requiring any further action by those developers. If there are any conflicts — if the two developers changed the same line of code, for example — then UCM would bring the conflicts to their attention so they could resolve it. The developer can instantly see all versions of the source code, including the original code, his changed versions, and any versions changed by other developers.

Once the work is delivered, the release engineer can now create a new baseline and route it to the testing group for final certification. A baseline is a reviewed and approved set of artifacts — including requirements, source code, test scripts, data files, design, and documentation — that constitutes a basis for further development. Developers can "rebase" their personal development streams and get updated source files from the new baseline, including the changes made by other developers on the team. UCM and ClearCase maintain a complete history of

each file, which developers and team leaders can view graphically in a detailed version tree (Figure 6.3). At ClassicsCD.com, Rational ClearCase and Rational ClearQuest are configured so that the "deliver" action automatically updates the enhancement request's status in Rational ClearQuest to "resolved". This streamlines the developer's activities, while providing the project manager with up-to-date, reliable information on the status of the request and the state of the project.

Summary of Key Rational ClearQuest Integrations

- Integration with Rational RequisitePro enables you to associate requirements to related enhancement requests or defects, ensuring that all stakeholder input is accurately communicated and addressed. And when Rational ClearCase is UCM-enabled, UCM requirements can be traced to the specific changes that implement them.
- Integration with Rational TestManager provides testers with the ability to enter defects directly from their testing environment. Additionally, those defects include relevant data that is critical for effective defect tracking and reproducing bugs. This integration completes a full-circle integration in Rational products, starting with entering an enhancement request in Rational ClearQuest, and then tracking the project requirements in RequisitePro, associating the requirements with test cases in TestManager, and preparing for the next iteration by creating defect records directly from failed tests.

Summary of Key Rational ClearCase Integrations

- Integration with Rational RequisitePro provides an efficient mechanism for requirements reuse and archiving. Using Rational ClearCase and Rational RequisitePro, you can capture a set of requirements at each iteration of the project, or at other periodic intervals.
- Integration with Rational Rose provides a managed, controlled environment for all model changes and powerful merge capabilities to simplify parallel development and increase team productivity. Rational Rose shows which models are checked out, while UCM integration tracks what model changes were made to complete each development activity.
- Integration with Rational TestManager supports baselining of test assets, allowing QA teams to capture a set of artifacts at each iteration of the project or at other predetermined times. This allows teams to "roll back" to a known state of the testing effort, so that they can effectively test a patch to a piece of software released some time ago. It also provides a reliable and simple mechanism for test asset reuse and archiving.

Automated Testing

If something is going wrong with the application your team is developing, when do you want to find out about it? If you are in denial, you might say, "Later" or even "Never". But if you want to get your project out the door on time, you need to know as soon as possible. Consider the plight of projects that follow the waterfall approach to development, advancing step-by-step from requirements analysis through design, coding and finally testing. Nobody can accurately predict how — or even if — the application will perform until the last stage of a long process. The project is at risk throughout the entire lifecycle because defects go undetected until the end. And defects found later in the development cycle are hundreds of times more difficult and costly to fix than they would be if found early on, particularly if repairing the defect will require substantial architectural changes.

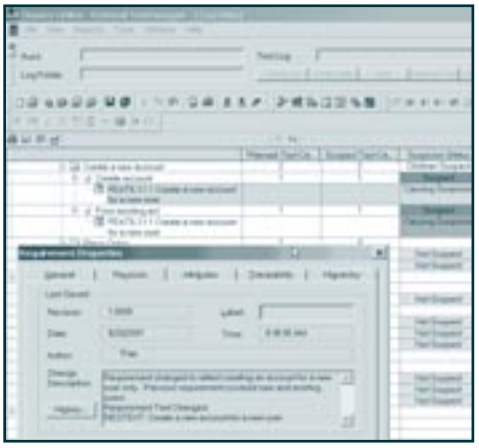

Figure 7.1: Rational RequisitePro and Rational TestManager. Within Rational TestManager, users can automatically see requirements properties from Rational RequisitePro.

The solution, of course, is to test as early and as frequently as possible within an iterative development process. Verifying software quality in this way offers several significant advantages:

- Project status assessment is made objective, not subjective, because test results can be tracked against defined requirements
- Objective assessment exposes inconsistencies in requirements, design, and implementations
- Testing can be focused on areas of highest risk, and
- Defects are identified early, reducing repair costs and minimizing their impact on the schedule

Testing early and often does mean additional work for the QA team. That is why teams equipped with Rational's automated testing tools have a significant advantage; they simply can get more testing done in less time. Further, testers must ensure not only that the system under test functions correctly, but also that it continues to function correctly under real-world user loads. And in many cases, using automated testing tools is the only practical way to accomplish this.

Automated Testing with Rational Suite TestStudio

Rational TestManager is an open and extensible environment that unites all of the tools, artifacts, and data both related to, and produced by your team's testing efforts: for functional, load, manual, integration, and unit testing. Rational TestManager manages all testing activities and artifacts throughout the iterative software development process. Test planning, design, scheduling, execu-

tion and results analysis are all made easier by Rational TestManager.

Rational Suite TestStudio includes the latest technology for Java and Web testing. RobotJ, for example, enables your team to create, modify, and run comprehensive automated tests for your applications. RobotJ uses Java as its scripting language; and its ScriptAssure™ technology minimizes script maintenance and automatically validates dynamic Web content.

Combined, Rational TestManager and RobotJ provide a powerful platform for automated functional and regression testing of Java and Web applications. Integration with Rational RequisitePro and Rational Rose allows your team to create and track tests for each and every requirement and use case. Also, through tight integration between Rational TestManager and Rational ClearQuest, testers can create new defect records directly from the test environment, and provide developers with the relevant information they need to reproduce the defect quickly.

Example Project: The Tester Sees New Requirements - Creates and Executes Tests

Back at ClassicsCD.com, now that the developer has delivered an activity to the main development stream, it is ready to be tested by the QA team. The QA team at ClassicsCD.com uses Rational TestManager as their primary desktop tool to direct testing activities. So first, in Rational TestManager a tester generates a requirements coverage report to see how many test cases have been implemented for each requirement. The structure of the test plan in Rational TestManager parallels the structure of the use-case requirements that were established in Rational RequisitePro, helping to ensure complete testing coverage of all the requirements in the ClassicsCD.com application. The tester sees a suspicion indicator, signaling him that a requirement associated with a particular test case has changed. The change, of course, is the new “Display Ship Date” requirement.

Testing Application Functionality Against Requirements

Using RobotJ, the tester:

- Records a new test script, and names it "Shipping Date"
- Inserts a verification point on the cashier page, and selects the shipping date text as the object to verify
- Stops recording and adds the new Shipping Date script to the test suite

Now whenever the test suite is run, the new script will verify that the “Display Ship Date” requirement is met. If, after a new build comes out, the ship date does not appear, Rational TestManager will flag the failure in its test results. The tester could then submit a defect to the project’s change request database to have engineering find out what is wrong, and fix it. When the tester uses TestManager to generate a defect, it fills in most of the information needed in the new record, and can also include specifics about the test run to help engineering reproduce the problem.

Ensuring Performance

Now that he has completed a functional test of the application, the tester can move ahead with another task that has been assigned to him by the project manager: ensuring the application will continue to work in the real world, where hundreds or thousands of visitors will access the ClassicsCD.com Web site simultaneously. The requirements for this project stipulate that every order-related transaction must occur in six seconds or less.

The tester uses Rational Suite TestStudio to validate that requirement by simulating realistic user loads with virtual users. To ensure performance, the tester:

- Records transactions between the browser and the Web server in virtual-user scripts
- Converts those single-user scripts into multi-user scripts by creating datapools, which will supply appropriate, varying data for things like credit-card numbers, customer logins, and more
- Graphically creates a realistic workload with several groups of users, all performing different transactions in the application
- Replays those transactions against the Web server, and records business-transaction performance and resource utilization metrics as hundreds of virtual users interact with the Web site
- Uses the extensive analysis and reporting capabilities in Rational TestManager to better understand and quantify the performance of the system as an end-user would see it

Seeing that the application meets its performance requirements, the tester and the project manager know that they can deliver the application, and be confident that the site can handle its expected workload.

Summary of Key Rational TestManager Integrations

- Integration with Rational RequisitePro and Rational Rose ensures that your QA team is always testing against the most up-to-date requirements. When requirements and models are used as test inputs for test cases, your team can develop and maintain a test plan in line with project requirements, and generate comprehensive test and model coverage reports. Through integration with Rational RequisitePro, changing a requirement automatically highlights any related test cases that might be affected.
- Integration with Rational ClearCase supports baselining of test assets, allowing QA teams to capture a set of artifacts at each iteration of the project, or at other periodic intervals. This allows teams to roll back to a known state of the testing effort, so that you can effectively test a patch to a piece of software released some time ago. It also provides a reliable and simple mechanism for test asset reuse and archiving.
- Integration with Rational ClearQuest enables testers to submit defect records directly from TestManager when viewing test results, and include pertinent data such as build, test script, and verification points to help developers reproduce the defect easily. And when the test

case is associated with a test input such as a requirement, the defect record created identifies the test input, providing full integration from feature inception to development and testing.

Summary of Key RobotJ Integrations

- Integration with Rational PureCoverage enables testers to collect valuable information about the quality of their software while executing automated scripts developed with RobotJ. For example, during regression tests, testers can also obtain detailed information regarding memory errors and leaks in code.
- Integration with Rational ClearCase allows version control of test scripts, eliminating the tedious and error-prone work of manual configuration management. This integration also enables your team to effectively manage concurrent changes to test assets.

Managing Project Risk with Rational

Today's project managers are being asked to guide the development of increasingly complex applications, with higher standards of quality, and on tighter schedules. In addition, often the expectation is that all of this will be accomplished on a tighter budget with fewer resources. In this environment, managing risk is more important than ever before in ensuring the success of your projects. To minimize risk and maximize predictability, many successful development teams have adopted Rational's iterative development approach and other best practices of software development. These best practices are fully supported by Rational's tightly integrated development tools that span every phase of the development lifecycle — from requirements and analysis, to visual modeling, software configuration management, and testing. This powerful combination of proven methodology and integrated tools helps teams increase productivity, efficiency, artifact reuse, and product quality, while minimizing chaos, miscommunication, rework, and ultimately, project risk.

Software development truly is full of risks. By providing seamless integration both internally and with popular developer tools and environments, Rational Suite simplifies the job of project managers. When everyone on your team can track requirements, enhancement requests, defect reports, test results, and other crucial project data, risk and uncertainty are minimized, and your entire development process becomes more predictable. All Rational Suite tools are updated, released, and installed together, so you spend less time maintaining your development environment and more time developing software. Each Rational Suite product combines powerful, integrated tools with proven best practices to help you manage risk and deliver more high-quality projects on time.

If you recognize the need to minimize risk and increase predictability in your organization's development projects, we encourage you to consider Rational's unique solution. We look forward to the opportunity to work with you.

About Rational Software Corporation

Rational Software provides a software development platform that improves the speed, quality, and predictability of software projects. This integrated, full lifecycle solution combines software engineering best practices, market-leading tools, and professional services. Ninety-eight of the Fortune 100 rely on Rational tools and services to build better software, faster. This open platform is extended by partners who provide more than 500 complementary products and services. Founded in 1981, Rational is one of the world's largest software companies, with revenues of $689.8 million in its twelve months ended March 31, 2002, and over 3,500 employees worldwide. Rational is a member of the S&P 500 Index and a component of the Nasdaq-100 Index®. Additional information is available at **www.rational.com** and **www.therationaledge.com**, the monthly e-zine for the Rational community.

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

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

Toll-free: 800-728-1212
E-mail: info@rational.com
Web: www.rational.com

International Locations:
www.rational.com/worldwide

TO800; rev. 8/02. Subject to change without notice.