A technical discussion of IBM Rational® ProjectConsole, a member of the IBM Rational® Team Unifying Platform
01/30/2004

IBM Rational Team Unifying Platform: IBM Rational® ProjectConsole Sample Measures

Bill Gilbert
Dave West
Doug T. Ishigaki

IBM Rational software White Paper

Table of Contents

Software & Systems Measurement with IBM Rational ProjectConsole

This white paper explains how progress and quality on an IBM Rational Suite project can be measured using IBM Rational® ProjectConsole. IBM Rational ProjectConsole, a member of the IBM Rational® Team Unifying Platform, can provide an entire development team and key stakeholders invested in the development of a software product and projects with on demand access to the information they need to keep a project on track from start to finish.

Although most everyone would agree that it's a great idea to track project measures throughout the product development lifecycle, in practice, most project teams haven't implemented measurement programs because of the high overhead involved. When a team is operating under tight time and budget constraints, gathering, analyzing, and disseminating data continuously are generally not perceived as high priority tasks. Unfortunately, without the power to assess project status and predict trends that reliable, up-to-date measures bring to a team, projects suffer.

The good news is that with Rational ProjectConsole, which comes bundled with all Rational Suite products, you can automate and unify your measurement collection, analysis, and display. With a minimal investment of time and effort on your team's part, ProjectConsole can collect data from the repositories of Rational Suite development tools and point products, as well as from third-party products such as Microsoft Project. It can display real-time results, in an easy-to-interpret graphical form, on a project Web site for access by an entire development team.

In this article, we provide some useful tips on how you can use ProjectConsole to easily put an effective measurement program in place. We describe a set of measurements a development team would find useful in monitoring a project through all the phases of a project developed using the Rational Unified Process®, or RUP® -- from inception to elaboration, through construction, and finally, transition.

There are more possible objects and attributes to measure in a software development project than we can (or care to) cover here. Most of the examples we show -- which represent just a small sample of the types of measures ProjectConsole can generate -- are prepackaged and available in the sample measures database that ships with Rational Suite products. They are based on data collected from IBM Rational® RequisitePro, IBM Rational® ClearQuest®, and IBM Rational® ClearCase®. IBM Rational Suite includes a tutorial that describes how to install and use ProjectConsole to deploy these measures into your production environment.

The measurement charts are designed to help project stakeholders determine how much work remains to be done in each phase of development; below, we'll suggest guidelines on what to look at during each phase of a RUP project and how you might interpret what you see.

When you first start using ProjectConsole to track your project's progress, it's a good idea to start with a small, simple set of measures that will satisfy immediate information needs in areas of concern in your project or organization. As you get comfortable with the technology, you'll want to add measures tailored to your project and its changing needs.

Monitoring the Inception Phase

The inception phase of a RUP project is focused on ensuring that all project stakeholders have a clear understanding of the scope of the project. Business and system requirements are of paramount importance.

The primary objectives of the inception phase are:

- Establish the project's scope and boundary conditions, including an operational vision, acceptance criteria and what is intended to be in the product and what is not
- Discriminate the critical use cases of the system, the primary scenarios of operations that will drive the major design trade-offs
- Exhibit, and maybe demonstrate, at least one candidate architecture against some of the primary scenarios
- Estimate the overall cost and schedule for the entire project (and more detailed estimates for the elaboration phase that will immediately follow)
- Estimate potential risks (the sources of unpredictability)
- Prepare the supporting environment for the project

Because of the emphasis of the inception phase is understanding requirements and determining risk, primary artifacts include:

- Vision of the project
- Initial Software Development Plan
- Business Case
- Initial Use Case Model
- Top-Level Class Diagrams
- Tailored Process Specification
- System Test Plan
- Risk list
- Risk Management Plan

Other artifacts may include:

- A Business Model
- An architectural proof of concept
- User Interface prototype
- Integration Build Plan

The inception phase is evaluated based on:

- Requirements understood, as evidenced of the fidelity of the critical use cases

- Completeness of the use case model reflecting vision features
- Risk identification and resolution strategies
- Sufficiency of software development plan to proceed to elaboration phase
- Stakeholder concurrence on scope definition and cost and schedule estimates
- Actual resource expenditures versus planned expenditures acceptable

Identifying business and system requirements are of paramount importance during Inception. Some of the identified requirements will be high level, and some will include more detailed descriptions. Throughout this phase, new requirements are added and refined. The charts in this section show data collected for the sample project over the first three weeks of an inception phase scheduled to last one month.

Monitoring total requirements

Figure 1 shows the number of project requirements collected from RequisitePro in the first three weeks of inception. Level 1 numbers represent high-level requirements, and Level 0 numbers represent detailed requirements.

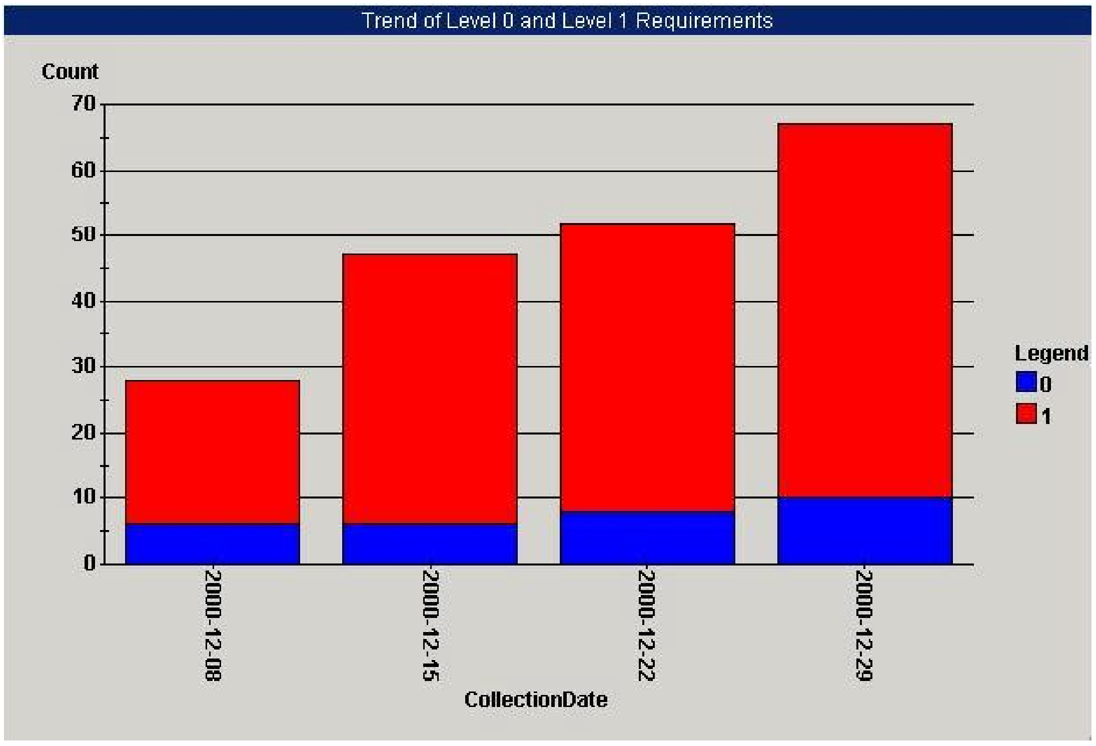

Figure 1: Project Requirements Collected During First Three Weeks of Inception

The measures depicted here indicate two noticeable trends. The first is that the total number of requirements is increasing. This suggests that project scope has not stabilized yet. Typically, the number of requirements stabilizes toward the end of inception, so these results indicate that the inception phase may have to be extended beyond the planned duration of one month.

The second noticeable trend is that, while the number of Level 1 requirements continues to increase, the number of Level 0 (high-level) requirements is stabilizing. This is what we'd expect to see, and it suggests that the project team has gained a good understanding of the project's defined scope. The high-level requirements are in place, with few new ones being added, and the requirements that flesh out the details are still coming in.

Assessing changes in requirement type

Figure 2 shows the number of use case requirements and features requirements defined for the project over the first three weeks of the inception phase, based on data collected from Rational RequisitePro. As expected, as the inception phase progresses and new requirements are identified, the numbers of new use cases and features are increasing.

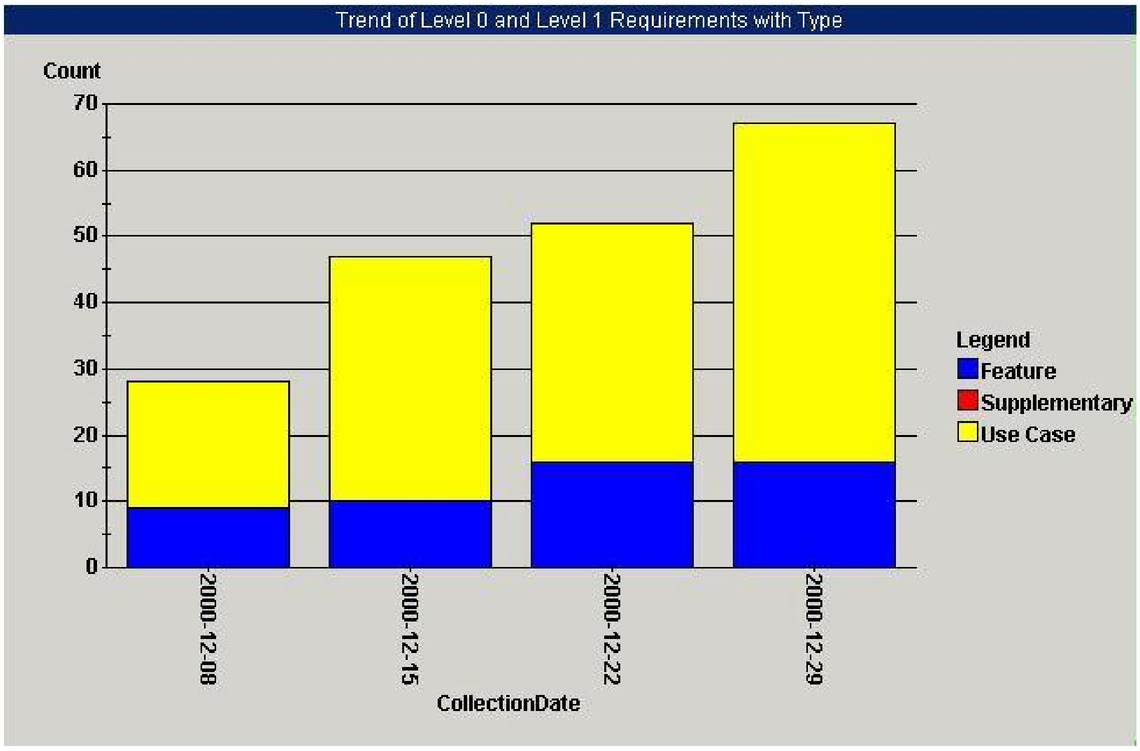

Figure 2: Use case and Features Requirements Defined During First Three Weeks of Inception

You can see evidence of two trends in this chart. First, the number of feature-related requirements is stabilizing, further supporting the impression that the team has nearly finished defining the project scope. The second trend is that the number of use case requirements is increasing. To complete the inception phase, the project scope must be stable. Although use case requirements will continue to grow throughout inception and elaboration, features should stabilize at the end of the inception phase.

Monitoring requirement status

During an iteration in a RUP project, the status of a requirement changes from *proposed* to *approved* to *incorporated* (if approved) and finally, to *validated*. At the beginning of the inception phase, the ratio of proposed requirements to approved requirements is high. By the end of inception, however, the situation should be reversed, with approved requirements far outnumbering proposed requirements. Figure 3 shows the number of proposed and approved requirements for the sample project in the first three weeks of inception, based on data collected from Rational RequisitePro.

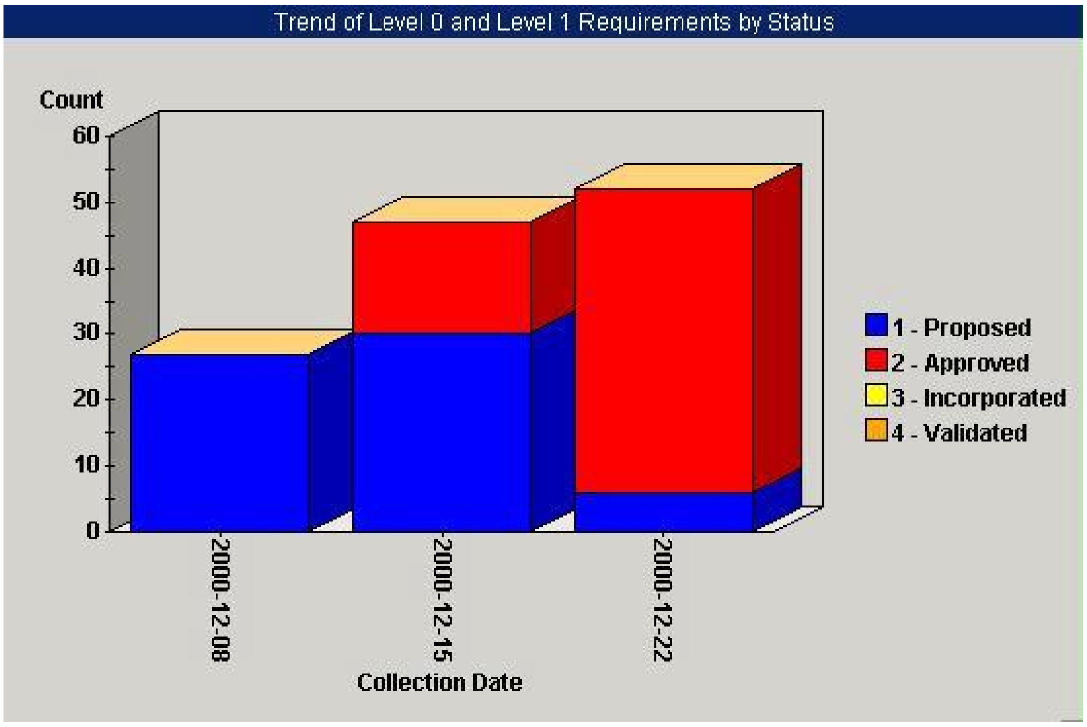

Figure 3: Proposed and Approved Requirements in the First Three Weeks of Inception

Figure 3 shows that, by the end of the third week, the vast majority of proposed requirements have been approved. This is a good sign that the project team is reaching consensus on project requirements. No prototyping has been done up to this point. This may be either a good or a bad thing, depending on the nature of the project. For some projects, the scope can’t be identified without some sort of prototyping; others, depending on scope and risk, may not need a prototype.

Monitoring requirements by priority

Toward the end of the inception phase, the team should have a clear understanding of the requirements, and information about requirement priority should be fairly well documented. Figure 4 shows the total number of requirements, based on their assigned project priority (1, 2, or 3).

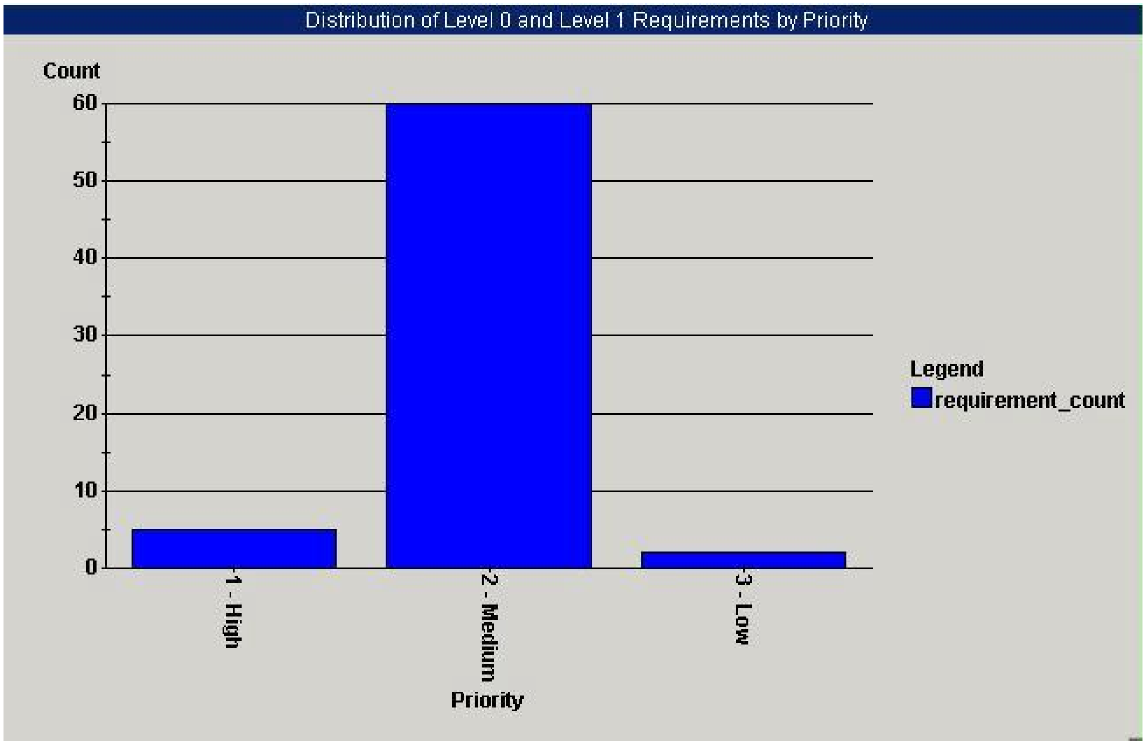

Figure 4: Total Number of Requirements, Based on Assigned Project Priority

At first glance, it's apparent that the distribution of requirements across the three priority groups is uneven. This may be an indication that no one has taken the time to make tough decisions about assigning priority. In this case, "medium" priority can be interpreted as "undecided."

Monitoring requirements by iteration

Toward the end of the inception phase, collective understanding of the requirements should be clearer, and information about planned iterations should be more complete. Figure 5 shows the proposed distribution of work for all requirements across planned iterations of development.

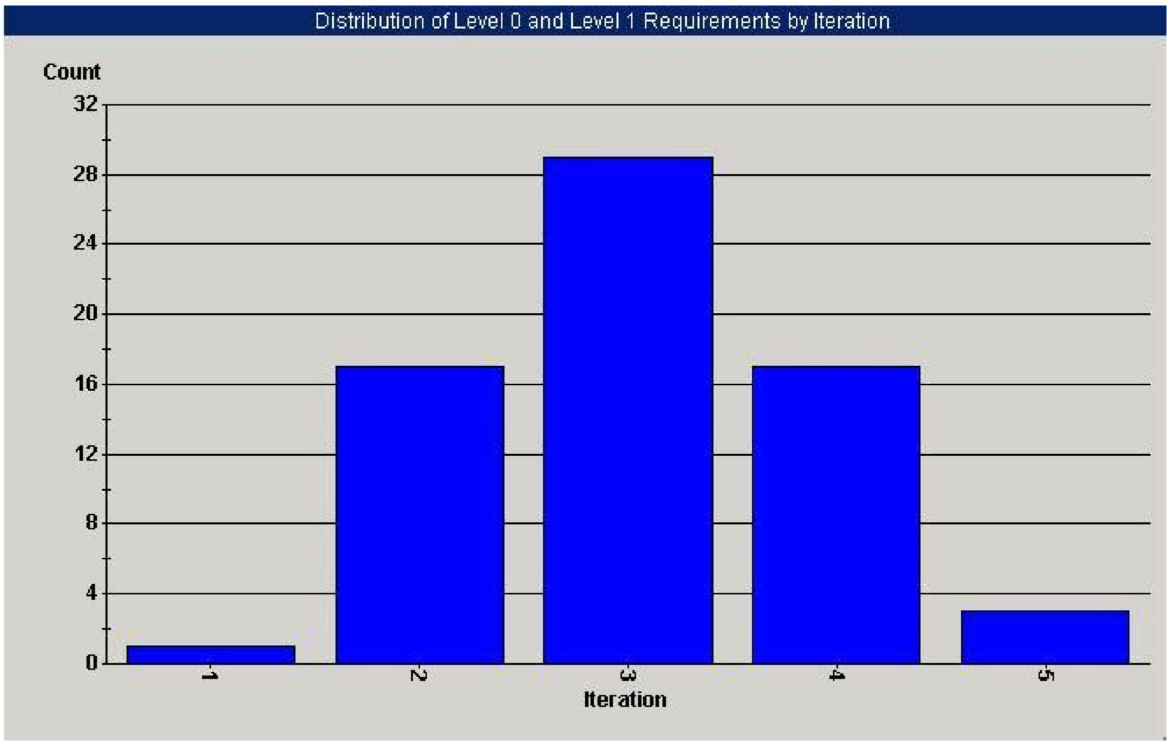

Figure 5: Distribution of Work for Requirements Across All Iterations

Figure 5 shows that the planned work is unevenly distributed across iterations. The relative sizes of the bars depend on the resources available for each iteration, the amount of work accomplished during previous iterations, and how difficult the requirements are to implement. In this case, we can see that there is a heavy load of requirements work toward the end of the elaboration phase, which may not be realistic.

Additional measures to monitor during inception

During the inception phase, you can use ProjectConsole to monitor the following measures:

Information Categories	Prospective Measures
Schedule and Progress	Task Completion Requirements Status Business Use Case Model Status Use Case Model Status Design Model Status
Resources and Cost	Staff Level, Turnover Effort, Earned Value – BCWS, BCWP, ACWP, SPI, CPI, SV, CV

Process Performance	Requirements-Design Traceability Requirements-Test Case Traceability Model Elements (e.g., Activity Diagrams)

Please note that the list of measures in this table is not meant to be exhaustive nor is it meant to be prescriptive. As we mentioned before, a successful measurement program monitors measurements that satisfy specific information needs in project areas of concern.

Monitoring the Elaboration Phase

During the elaboration phase of a RUP project, the development team focuses on clarifying the architecture's scope, major functionality, and nonfunctional requirements such as performance requirements.

The primary objectives of the elaboration phase are:

- Ensure that the architecture, requirements, and plans are stable enough
- Establish and demonstrate a sound architectural foundation
- Analyze the problem domain
- Design the solution
- Address all architecturally significant risks of the project
- Develop a comprehensive plan for the construction and transition of the project
- Demonstrate that the baselined architecture will support the requirements of the system at a reasonable cost and in a reasonable time
- Refine previous course-grained plans

Artifacts of the elaboration phase include:

- Elaborated and/or baselined Vision (requirements)
- Detailed Use Case Model
- Detailed Class Diagrams
- Software Architecture Document
- Executable Baseline Architecture
- Revised Software Development Plan
- Risk list

The elaboration phase is evaluated based on:

- Stability of the product vision (requirements)
- Stability of the architecture
- Resolution of risks
- Sufficiency and credibility of the plan for the construction phase
- Stakeholder buy-in that the current vision can be met if the current plan is executed to develop the complete system in the context of the current architecture
- Actual expenditures versus planned expenditures acceptable

The sample charts in this section show data collected for a sample project over the first three weeks of an elaboration phase scheduled to last one month.

Examining use cases that affect system architecture

One effective way to assess the progress of the elaboration phase is to check the status of use cases that directly affect the project's architectural design. By the end of the elaboration phase, all of these use cases should be verified. Figure 6 shows the total number of use cases, classified according to whether they do or do not affect the system architecture, three weeks into the elaboration phase.

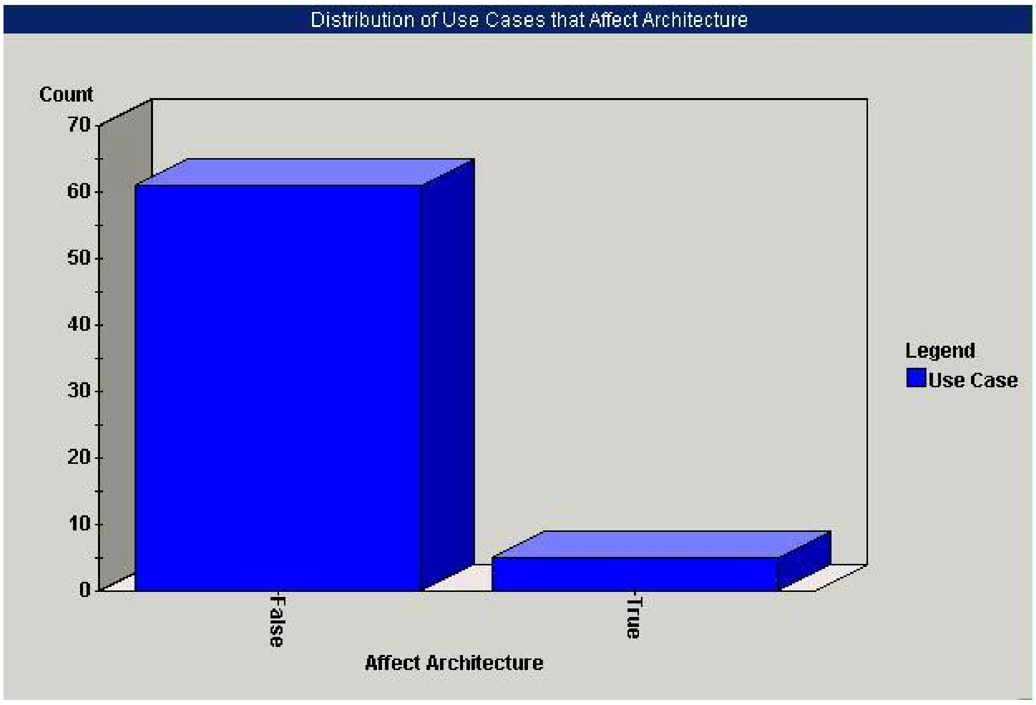

Figure 6: Number of Use Cases Three Weeks into Elaboration

Figure 6 shows that just five use cases require validation during elaboration. If no use cases had been developed, it would not be possible to complete the elaboration milestone and exit that phase.

Distribution of affects-architecture use cases by status

Figure 7 shows the chart that's displayed if a ProjectConsole user "drills down" to see the details for the five use cases that affect the architecture. (To drill down for these details, a user can double-click the "True" column in the chart.)

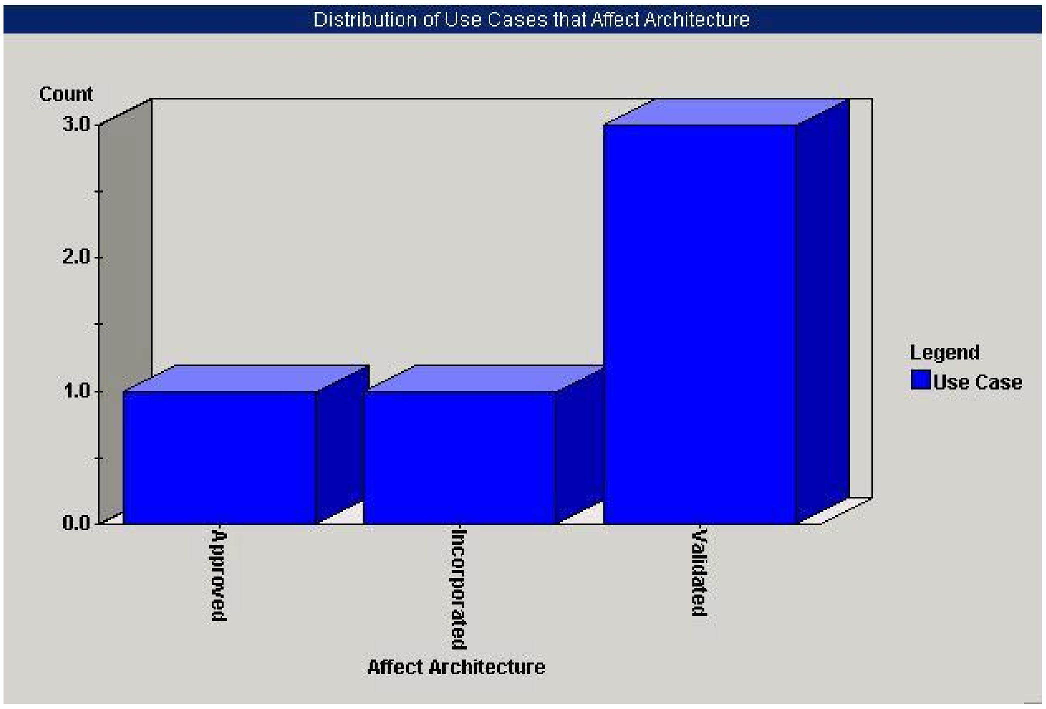

Figure 7: Detail for Elaboration Use Cases

Figure 7 shows that only two use cases remain to be validated before the end of elaboration. One of these has already been incorporated, and the other has yet to be incorporated. We can see that, from an architectural design standpoint, some work is required before the elaboration phase can come to a close.

Assessing the level of detail in requirements

As elaboration progresses, more requirements are defined in greater detail. As elaboration nears a close, all high-level requirements should be fleshed out in preparation for the next phase - construction. Figure 8 shows the distribution of requirements, based on level of detail, two weeks into the elaboration phase. In this chart, the detail level increases with the number. A zero represents almost no detail, and a three represents the highest level of detail.

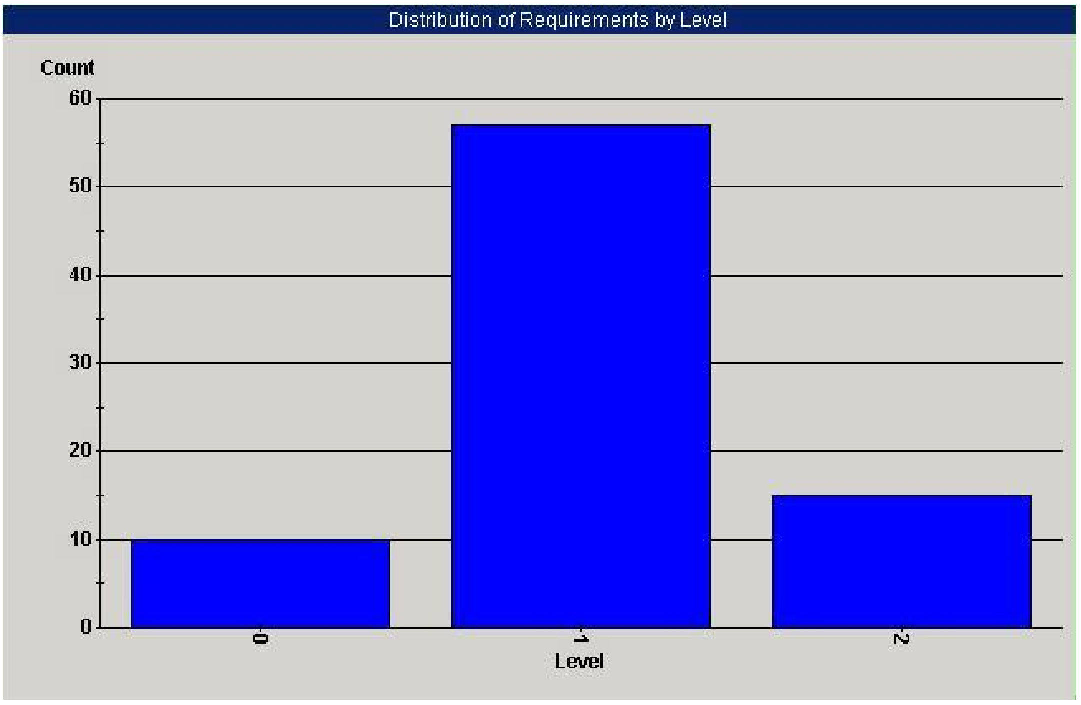

Figure 8: Distribution of Requirements, Based on Level of Detail, Two Weeks into Elaboration

Halfway through the elaboration phase, a higher proportion of requirements should be classified as “2”s, and few or none should be classified as “0”s. The results shown in Figure 8 suggest that more requirements need to be fleshed out, and that some prototyping might be needed.

Monitoring requirements by status

Towards the end of elaboration, requirements should begin to be incorporated and the number of Approved vs. Incorporated requirements should begin to stabilize. There should be an upward trend of more Incorporated requirements as we’re gaining consensus on the requirements.

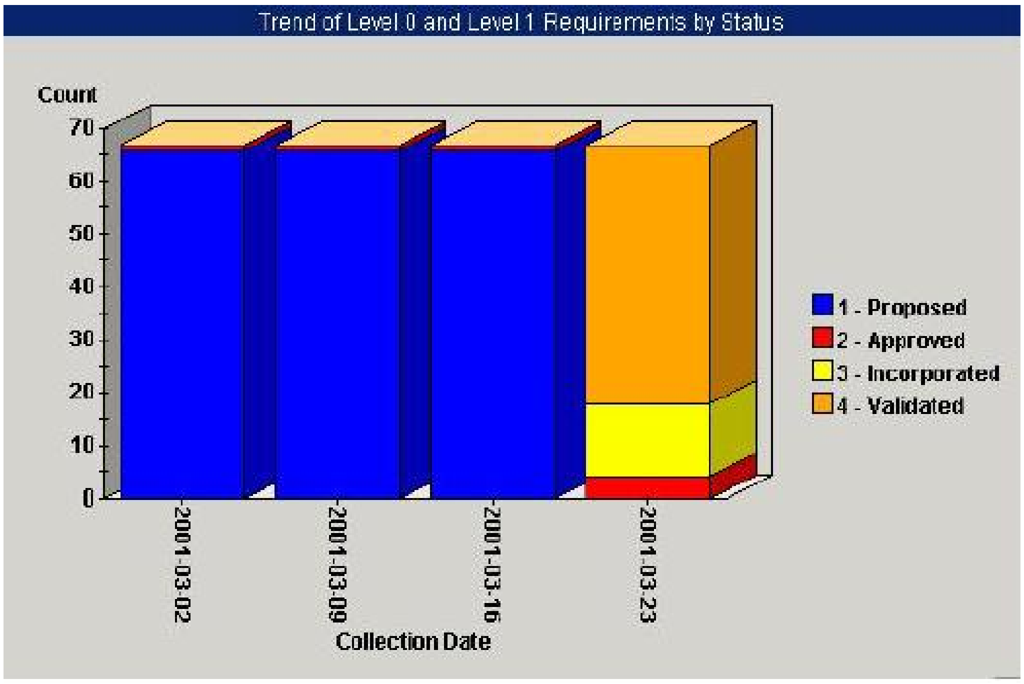

Figure 9: Trend of Requirements by Status

Figure 9 shows a trend that should raise questions during elaboration. First of all, a project manager would expect that the high level of proposed requirements means that this project probably was not ready to transition out of the inception phase to the construction phase. Second, there should be an steady upward trend on the number of requirements in the Approved state. The abrupt rise in Incorporated and Validated in the last week should signal the project manager to question whether the requirements analysis process is being followed.

Monitoring requirements by iteration

In an iterative development project, work is performed incrementally in iterations. The goal for planning should be to have an even distribution of requirements that will be worked on in each iteration.

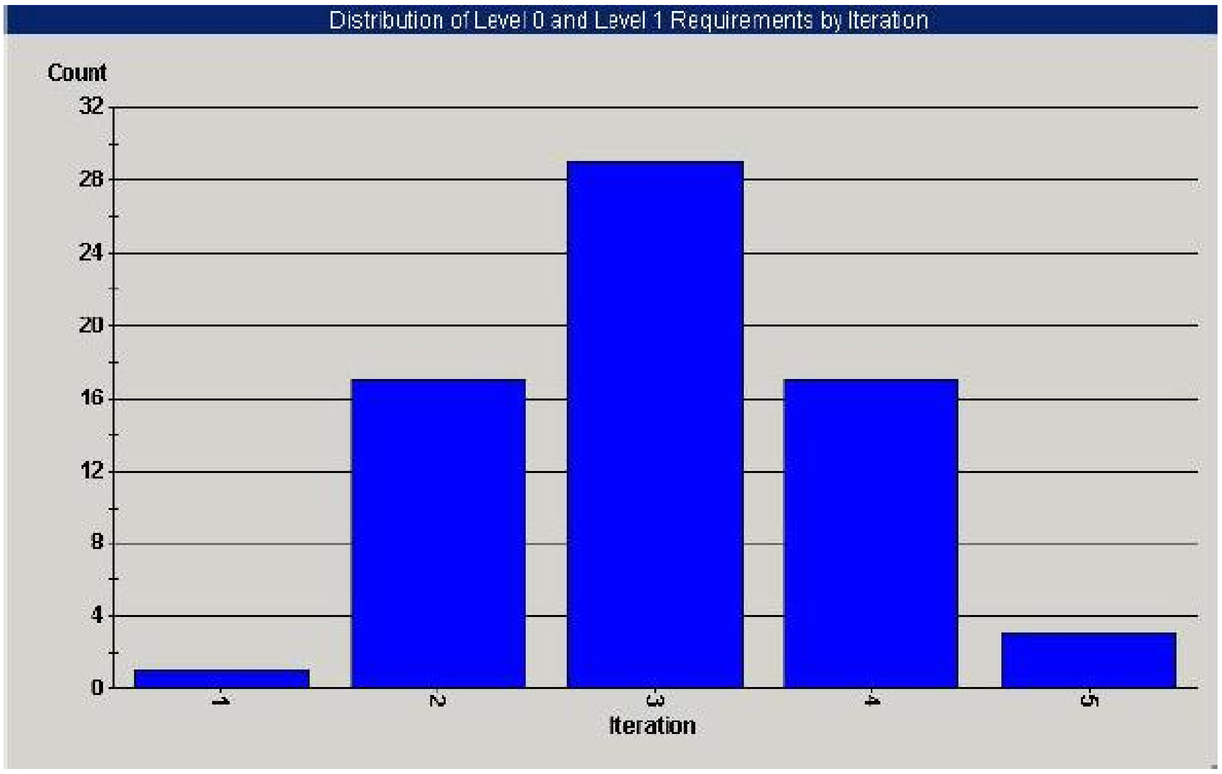

Figure 10: Distribution of Requirements by Iteration

Figure 10 shows that the planned work isn't evenly distributed across the iterations. Therefore, this indicator should signal the project manager to revisit the iteration plan.

Additional measures to monitor during elaboration

During the elaboration phase, you can also use ProjectConsole measurement charts to further monitor the following measures:

Information Categories	Prospective Measures
Schedule and Progress	Task Completion Requirements Status Requirements Tested Use Case Model Status Design Model Status Units Designed, Coded, Tested Test Cases Attempted, Passed, Failed
Resources and Cost	Staff Level, Turnover Effort, Earned Value – BCWS, BCWP, ACWP, SPI, CPI, SV, CV
Product Size and Stability	Requirements, Requirements Volatility (Churn) Lines of Code Function Points

	Components
	Interfaces
Product Quality	Defects
Process Performance	Requirements/Model Traceability
	Requirements/Test Case Traceability
Technology Effectiveness	Requirements Coverage

Monitoring the Construction Phase

During the construction phase of a RUP project, a project team focuses on developing and integrating all components and features into the product, and testing these features thoroughly.

The primary objectives of the construction phase are:

- Minimize development costs by optimizing resources and avoiding unnecessary scrap and rework
- Achieve adequate quality as rapidly as practical
- Achieve useful alpha, beta, and other test release versions as rapidly as rapidly as practical
- Complete the analysis, design, development, and testing of all required functionality
- Iteratively and incrementally develop a complete product that is ready to transition to its user community, including describing remaining use case requirements, fleshing out the design, completing the implementation, and testing the software
- Decide if the software, sites, and the users are all ready for the application to be deployed
- Achieve some degree of parallelism between teams (resource permitting) since parallelism acan accelerate development activities

Artifacts in the construction phase include:

- Executable releases of increasing functionality
- Models of the system's design and behavior
- User documentation
- Deployment documentation
- Evaluation criteria fore each iteration
- Release descriptions, including quality assurance results
- Updated Software Development Plan
- Integration Build Plan

Evaluation of the construction phase is based on:

- Product baseline maturity and stability; pending changes and existing defects are not obstacles for achieving a release
- Readiness of stakeholders to transition the product to the user community
- Actual resources expenditures versus planned expenditures acceptable

The charts in this section show data collected for the sample project during a construction phase scheduled to last three months.

Monitoring open defects

Figure 11 shows the number of open defects over a seven-week period based on data collected from Rational ClearQuest.

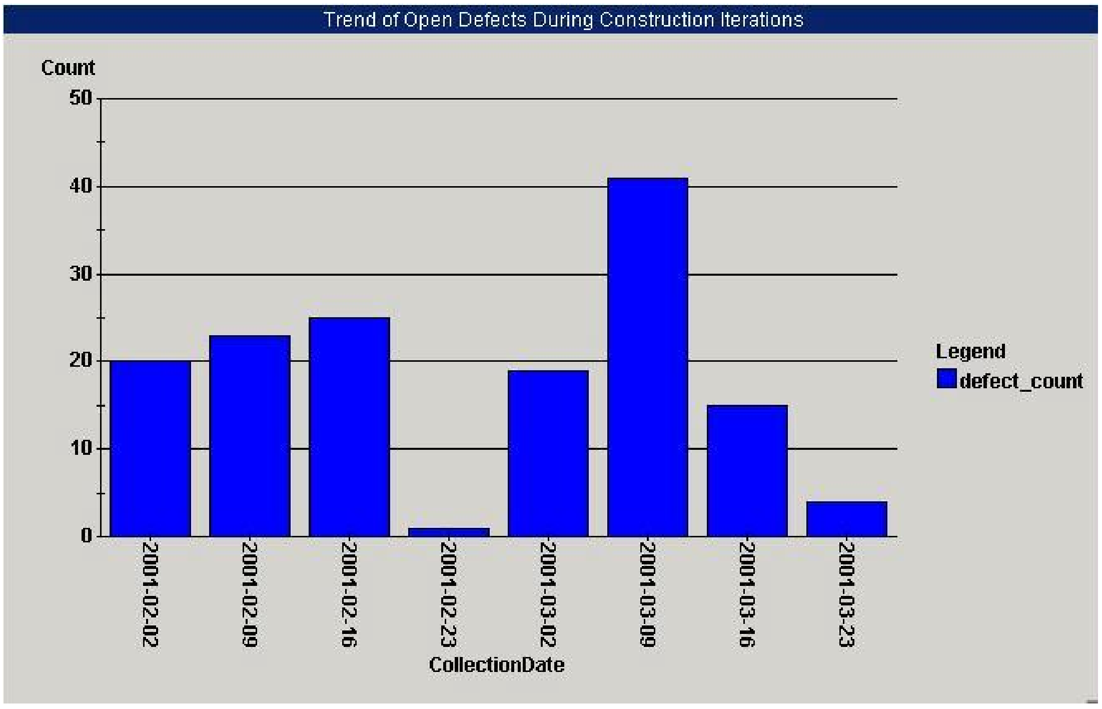

Figure 11: Open Defects Over a Seven-Week Period During Construction

One of the trends shown in Figure 11 approximates what you expect and want to see during the construction phase - periods during which the number of open defects is high, followed by a dramatic decline in that number. This pattern suggests that the development team is stabilizing the product at the end of each construction iteration.

Another noticeable trend is a continuous increase in the number of open defects, which probably means that the testing program in place is effective.

Monitoring lines of code added, modified, and deleted

Figure 12 shows the code churn over an iteration in the construction phase. The data for this measure was collected from Rational ClearCase.

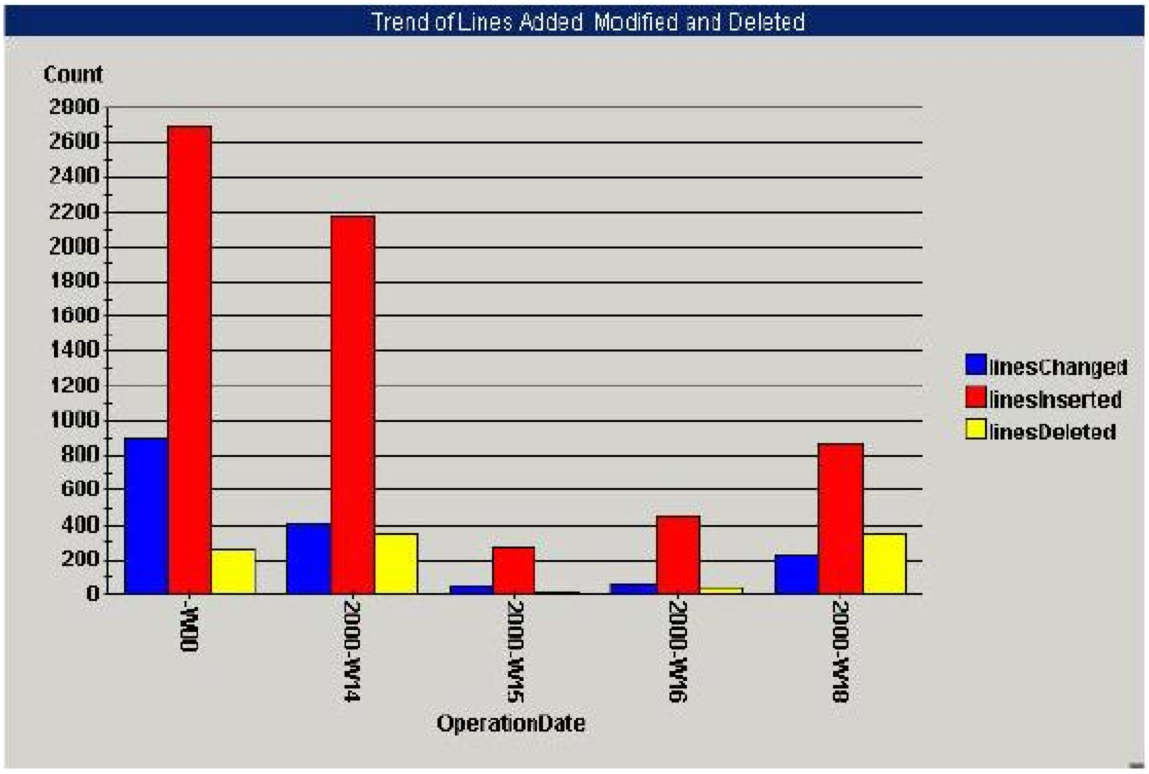

Figure 12: Trend of Lines Added, Modified, Deleted over Construction

In this trend, lots of code was added at the beginning compared to the end. This would be a good trend - indicating the code base was stabilizing. However, the trend also is showing a slight increase in the churn near the end. This may indicate problems. The increase in code added, modified, and deleted may indicate that testing is uncovering defects and the product may not be ready to transition into the next iteration.

Monitoring use case status

Figure 13 shows a trend of the number of approved, incorporated, and validated use cases to date.

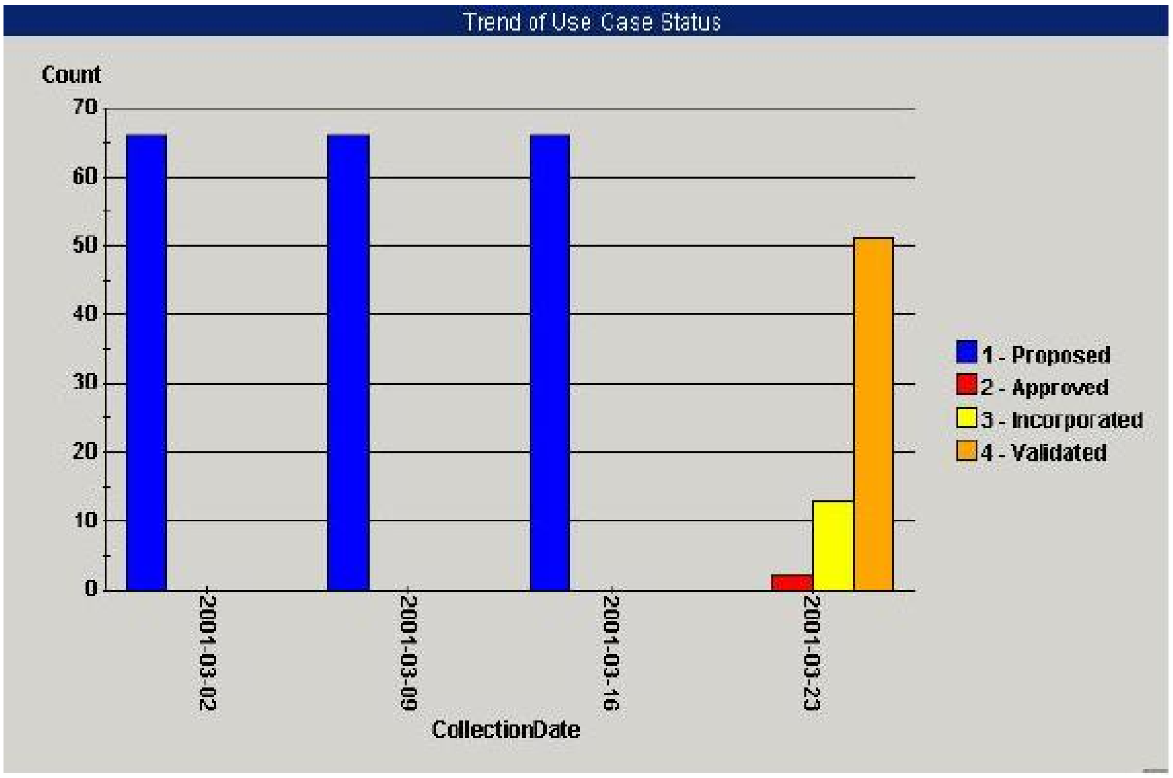

Figure 13: Trend of Use Case Status over Construction

This trend indicates that validation of use cases isn't happening until the end. This indicates a waterfall development process rather than an iterative process approach. Or perhaps the testing process is not reporting progress until the end. In either case, the project manager needs to make sure that enough test resources have been identified early and that testing is performed throughout the construction phase.

Additional measures to monitor during construction

Throughout the construction phase, you can use ProjectConsole to monitor the following measures:

Information Categories	Prospective Measures
Schedule and Progress	Task Completion Change Requests Opened, Resolved Units Designed, Coded, Tested Test Cases Attempted, Passed, Failed
Resources and Cost	Staff Level, Turnover Effort, Earned Value – BCWS, BCWP, ACWP, SPI, CPI, SV, CV
Product Size and Stability	Requirements, Requirements Volatility (Churn) Lines of Code, Code Churn Function Points Components Interfaces
Product Quality	Defects

	Age of Defects
	Cyclomatic Complexity
	Mean-Time-to-Failure
Process Performance	Defects Contained
	Defects Escaping
	Scrap, Rework Effort
	Requirements/Model Traceability
	Requirements/Test Case Traceability
	Change Request/Test Case Traceability
Technology Effectiveness	Requirements Coverage
Customer Satisfaction	Customer Reported Defects

Monitoring the Transition Phase

In the transition phase of a RUP project, the development team focuses on moving the software product to the user community.

The primary objectives of the transition phase are:

- Achieve user self-supportability
- Achieve stakeholder concurrence that deployment baselines are complete and consistent with the evaluation criteria of the vision
- Achieve final product baseline as rapidly and cost effectively as practical

The artifacts of the transition phase are:

- Executable releases
- Updated system models
- Release descriptions, including quality assurance results
- Updated user manuals
- Updated deployment documentation
- Training materials
- Project Close-Out Plan portion of the Software Development Plan

The success of the transition phase is evaluated based on user satisfaction with previous product versions and the balance between the actual resources expended versus planned expenditure.

The charts in this section show data collected for the sample project during a transition phase scheduled to last one month.

Monitoring total defects for the release

The following chart (Figure 14) shows the number of total defects over the entire development lifecycle.

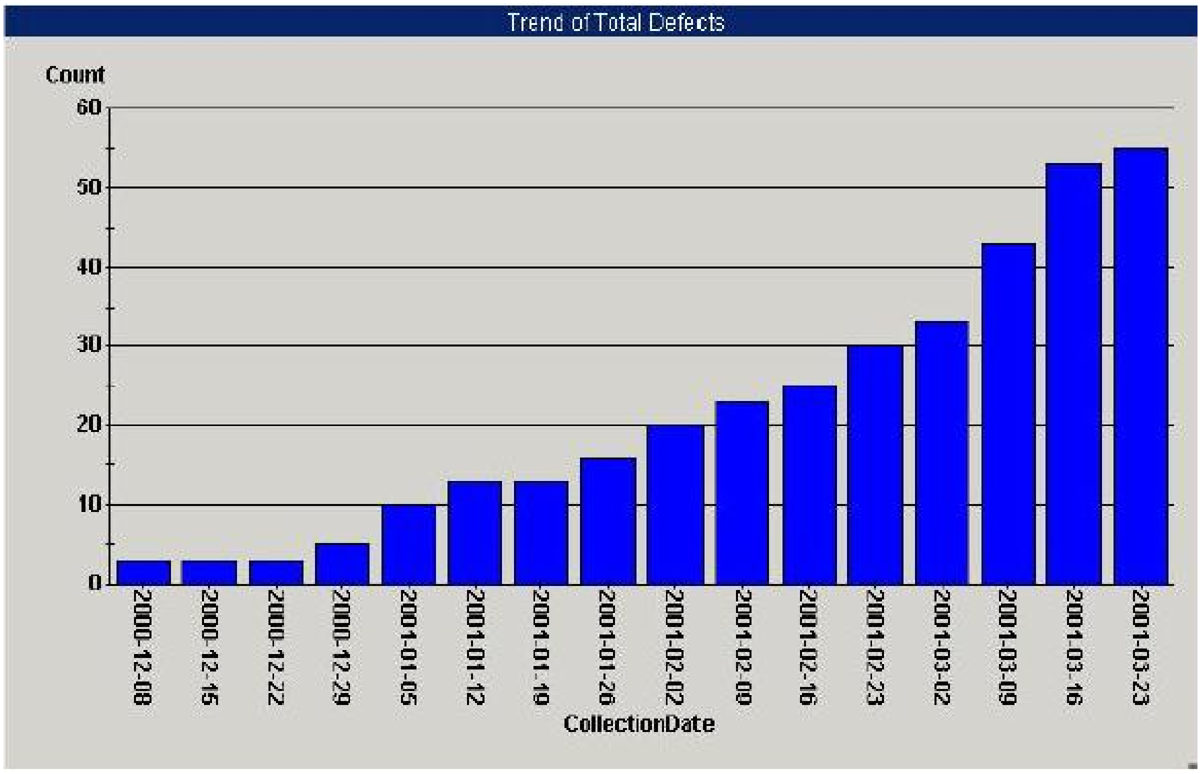

Figure 14: Trend of Total Defects for the Release

The number of total defects across the entire development has leveled off, which suggests that the product has stabilized.

Monitoring open defects for the release

Figure 15 shows the number of open defects over the entire development lifecycle.

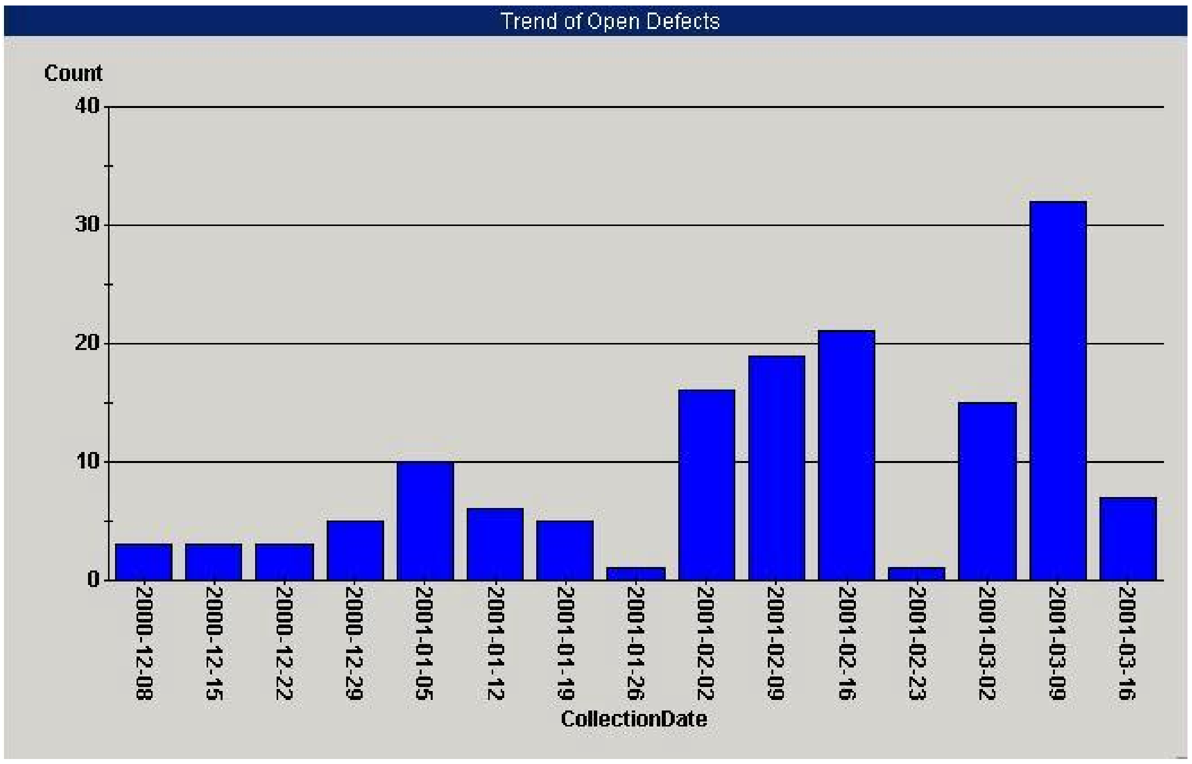

Figure 15: Trend of Open Defects for the Release

A given release can include multiple iterations. At the beginning of each iteration, the trend of open defects may increase, but as you near the end of each, and the release milestone, the number of open defects should decrease, indicating that the quality of the product is reaching an acceptable state.

Monitoring defect status in this release

Figure 16 shows a trend of all defects by status in this release.

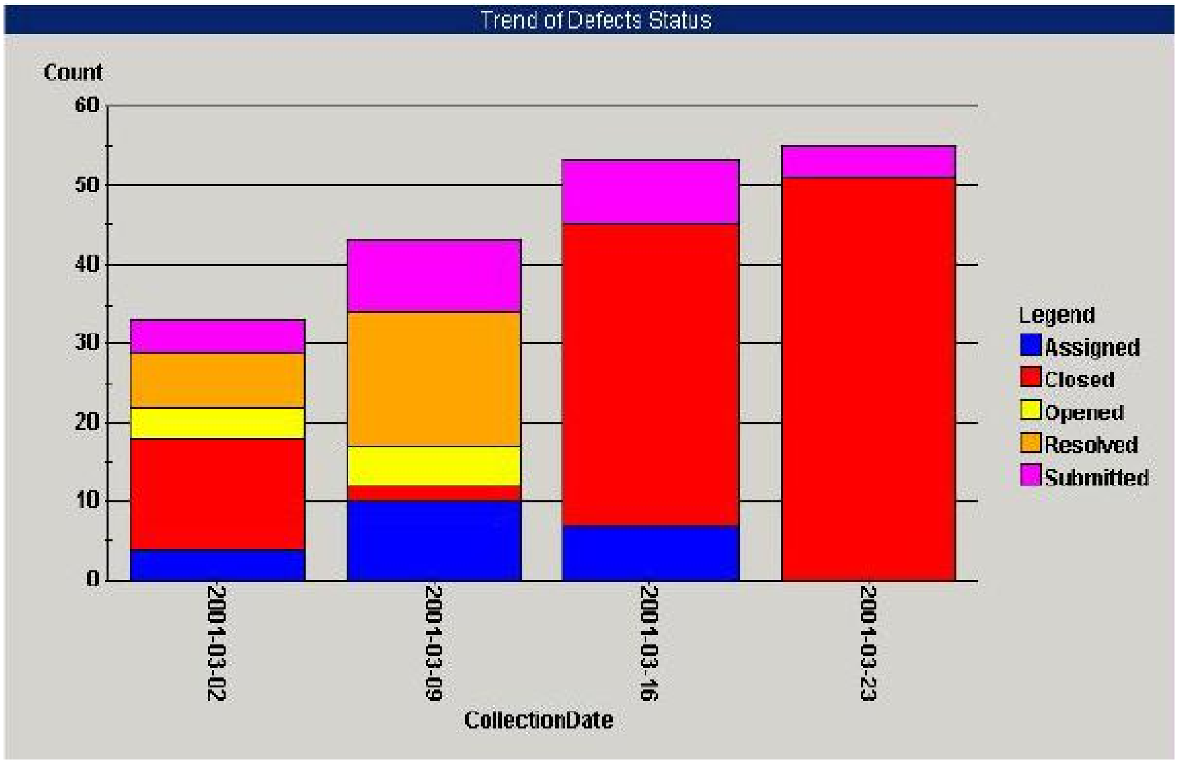

Figure 16: Trend of Defect Status

As the project is nearing completion, the majority of identified defects should be closed. This indicates that the product quality is at a state where the product is ready for release.

Monitoring lines added, modified, and deleted

Figure 17 shows the code churn during the transition phase.

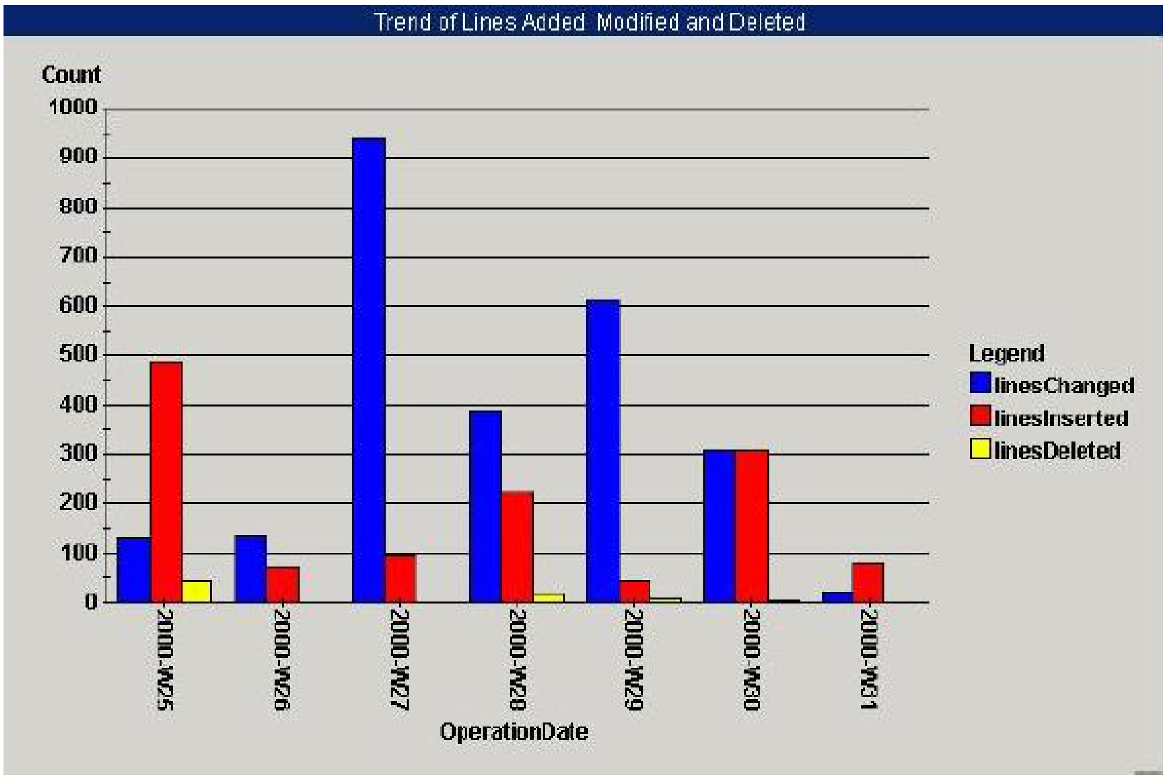

Figure 17: Trend of Lines Added, Modified, and Deleted During the Phase

Code churn (lines added, modified, and deleted) has decreased to very low levels, supporting the impression the product is almost ready for release. Ideally, there should be no code churn at all near the release milestone. If there is some amount of code churn, there is a risk that quality problems may have been introduced and additional regression testing may need to be performed before the product is ready for release.

Additional measures to monitor during transition

Throughout the transition phase, you can use ProjectConsole measurement charts to monitor the following measures:

Information Categories	Prospective Measures
Schedule and Progress	Task Completion Change Requests Opened, Resolved Test Case Progress
Resources and Cost	Effort, Earned Value – BCWS, BCWP, ACWP, SPI, CPI, SV, CV
Product Quality	Defects, Code Churn
Customer Satisfaction	Customer Reported Defects

Summary

In this white paper, we've introduced you to simple measurements that you can use to measure the progress and assess quality during project development using the Rational Unified Process and Rational Suite. The sample metric charts were produced using Rational ProjectConsole.

ProjectConsole allows a software development team to automatically quantify the current project status and assess development trends of their project with up-to-date measures. ProjectConsole collects measurement data on a specified scheduled or on demand, from the Rational Suite's development environment and selected third-party tools. The results are visually presented in graphs, charts and gauges.

With ProjectConsole charts and indicators, all project team members can analyze low-level details, planned-versus-actual measures, historical data, and trend charts to get an overview of an entire project. This information enables a software development team to set realistic project expectations, more realistically assess potential risks, identify bottlenecks, realize the cause for late deliverables, take prompt corrective action, forecast future project milestones, and ultimately, put the entire team in a better position to objectively and accurately measure project progress and quality.

About IBM Rational software

IBM Rational software, formerly an independent company and now one of the IBM software brands, offers a comprehensive software development solution. The IBM Rational software platform combines software engineering best practices, market-leading tools, and expert professional services, all of which drive rapid and continuous improvement in software development capability for on demand businesses.

In addition, IBM Rational software offers more than 20 years experience in promoting and delivering integrated and open software systems, both of which are key characteristics of the on demand operating environment.

Integrated – IBM Rational software has contributed considerable thought leadership and expertise in the areas of Service-Oriented Architecture (SOA), enterprise and software architecture, and heterogeneous platform support.

Open – IBM Rational software has a long history in developing and supporting the goals of open computing. This includes development of the Unified Modeling Language (UML), now a standard for modeling applications, database designs, and business processes. IBM Rational software has promoted and participated in the development of a wide variety of open computing standards. It offers support for major programming languages and operating platforms, and it provides an extensive set of application programming interfaces for third-party tools interoperation.

Thousands of companies around the world have realized the benefits of the approach advocated by IBM Rational software. Their processes are results-oriented, the artifacts they produce are well designed and reusable, and they are working at higher levels of capability now required by the on demand era.

Additional References

1. Royce, W., Software Project Management A Unified Framework, Addison Wesley, 1998
2. Kruchten, Philippe. *The Rational Unified Process, An Introduction*, Second Edition, Reading, MA: Addison-Wesley, 2000.
3. Ishigaki, Doug, Jones, Cheryl, *Practical Measurement in the Rational Unified Process*, the Rational Edge, January 2003 Edition.
4. McGarry, John, David Card, Cheryl Jones, Beth Layman, Elizabeth Clark, Joseph Dean, and Fred Hall. *Practical Software Measurement – Objective Information for Decision Makers*, Boston, MA: Addison-Wesley, 2002.
5. *ISO/IEC 15939 Software Measurement Process*. 2002.
6. Software Engineering Institute, “Capability Maturity Model® Integrated (CMMI(sm)) for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing Version 1.1 Staged Representation.” Pittsburgh: Carnegie Mellon University, Mar. 2002.
7. IBM Rational ProjectConsole, Version 2003.06.00
8. IBM Rational Unified Process, Version 2003.06.00.
9. IBM Rational Team Unifying Platform, Version 2003.06.00.

IBM software integrated solutions

IBM Rational supports a wealth of other offerings from IBM software. IBM software solutions can give you the power to achieve your priority business and IT goals.

DB2® software helps you leverage information with solutions for data enablement, data management, and data distribution.

Lotus® software helps your staff be productive with solutions for authoring, managing, communicating, and sharing knowledge.

Tivoli® software helps you manage the technology that runs your e-business infrastructure.

WebSphere® software helps you extend your existing business-critical process to the Web.

Rational® software helps you improve your software development capability with tools, services, and best practices.

Rational software from IBM

Rational software from IBM helps organizations create business value by improving their software development capability. The Rational software development platform integrates software engineering best practices, tools, and services. With it, organizations thrive in an on demand world by being more responsive, resilient, and focused. Rational's standards-based, cross-platform solution helps software development teams create and extend business applications, embedded systems and software products. Ninety-eight of the Fortune 100 reply on Rational tools to build better software, faster. Additional information is available at **www.rational.com** and **www.therationaledge.com**, the monthly e-zine for the Rational community.

IBM Corporation
Software Group
Route 100
Somers, NY 10589
U.S.A.

Printed in the United States of America 05-03
All Rights Reserved. Made in the U.S.A.

The IBM home page on the Internet can be found at ibm.com