Test analysis and reporting using IBM Rational Quality Manager

Michael Kelly

December 16, 2008

This article gives you an in-depth look at test analysis and reporting using IBM® Rational® Quality Manager and covers some of the common questions that a test manager might ask. You will also learn how to use the data provided to assist in qualitative and quantitative analysis for your testing projects.

Author's note:

This article was written using IBM® Rational® Quality Manager Open Beta Version 1.0.0.20080919_1011 running on a Microsoft® Windows® XP Professional SP3 server, using a Microsoft Windows 64-bit Vista™ Business SP1 with Mozilla FireFox 3.0.2 browser. In addition, this article references www.BookPool.com, the version that was publicly available at the time of writing.

IBM® Rational® Quality Manager is collaborative, Web-based, quality management software that offers comprehensive test planning and test asset management for the full software development life cycle. Built on the Jazz™ platform, Rational Quality Manager is designed to be used by test teams of all sizes. It supports a variety of user roles, such as test manager, test architect, test lead, tester, and lab manager, as well as roles outside of the test organization.

This article gives you an in-depth look at test analysis and reporting using Rational Quality Manager and covers some of the common questions that a test manager might ask. You will also learn how to use the data provided to assist in qualitative and quantitative analysis for your testing projects.

Planning for reporting

As with any tool, the information that you get from Rational Quality Manager is only as good as what you put into it. When you think about your company, your project, and what information you need to report, also think about how you need to set up and use Rational Quality Manager to get that information. Ask yourself these questions:

- How much testing do we plan to do and how much of that have we done?
- What potential tests are remaining and in which areas?
• What is our current test *creation* velocity and what has it been for the duration of the project? How does that break out across the testing area or test case priority?

• What is our current test *execution* velocity and what has it been for the duration of the project? How does that break out across the testing area or test case priority?

• How much of our testing has been focused on basic functionality (basic requirements, major functions, simple data) rather than common cases (users, scenarios, basic data, state, and error coverage) rather than stress tests (strong data, state, error coverage, load, and constrained environments)?

• How much of our testing has been focused on capability rather than other types of quality criteria (performance, security, scalability, testability, maintainability, and so on)?

• How many of the requirements have we covered and how much of the code have we covered?

• If applicable, what platforms and configurations have we covered?

• How many issues have we found, what severity are they, where have we found them, and how did we find them?

Rational Quality Manager will not answer all of those questions, but it can address many of them, and it can provide some of the information required to answer others.

**Set up and manage your requirements**

In many companies, requirements coverage or requirements traceability is a big focus of the testing process. Sometimes, it's support an application development requirement in a regulated industry. Other times, it is one metric for determining whether the team has done enough testing. Regardless of why you are interested in requirements coverage, if it's something that you hope Rational Quality Manager can help with, then you have setup work that you'll need to do.

**Managing and tracing requirements**

If your requirements are managed in an external tool, such as IBM® Rational® RequisitePro®, then you can link the requirements into the test plan. If you are not using an external tool, then you need to manage the requirements in the test plan. Regardless of how it happens, after you have entered the requirements, you can associate them with test cases to trace your test scripts all the way back to your requirements.

When requirements are changed or deleted, the status of the requirement in Rational Quality Manager is updated to show the latest status. Test cases that contain requirements that change or are deleted are flagged so that you can adjust test plans and test cases quickly and accurately to respond to requirement changes.
Rating severity
When you have numerous tests for even the smallest of functions within an application, it can be difficult to explain to a project team what kind of coverage you're getting. Even when talking about requirements coverage (one possible measurement of many), it can be helpful to know which requirements are more important than others so you can plan your testing appropriately. Higher-severity requirement might warrant more test cases, or reviews by more testers, or more detailed test documentation.

Knowing the severity of a requirement allows you to answer certain questions, for example:

- What percentage of the high-severity requirements have I covered relative to the low-severity requirements?
- How many test cases do I have for those requirements relative to other areas?
- Where am I finding defects? (In high-severity-requirement test cases or somewhere else?)

Upfront, you might want to take some time to define a severity scheme for your requirements and then try to make sure that you follow it. This is useful not only in making sure that you are focused on the right things, but also when you consider ideas for continuous improvement after completing the project. Figure 1 shows an example of setting severity level for a requirement.

Using tags
In addition to severity, each requirement includes a Tags field. Tags are keywords that can help you manage your testing. You can use them to classify requirements, test cases, and defects with different areas of the application and different types of testing. For example (regardless of tooling):

- **Areas of the application**: core platform, reporting, client-facing, internal tools, external tools, voice, and so forth
- **Quality criteria**: capability, usability, security, performance, compatibility, testability, supportability, and so on

With Rational Quality Manager, you can tag requirements with your own keywords. That enables you to generate reports to answer questions such as these:

- How many test cases are for reporting (or another area)?
- How many requirements stipulate security testing (or another quality criteria)?
- What percentage of requirements are focused on capability vs. security vs. performance?

This shows you where you are spending your testing time and what types of concerns you are looking for. You can see an example of defining tags for a requirement in Figure 1.
Set up and manage your test cases

Another key area where you can do a little upfront work to get big dividends later is in your test cases. There are two key areas in test cases that can help you manage your testing:

- **Test case weight**, which allows for more granular and honest results reporting, thus increasing the value of your data results
- **Test case categories**, which, like tags, enable you to partition your test cases for more detailed reporting

Specifying test case weight

When you create a test case, you have the option to assign a test case weight to it. The idea behind this is that not all test cases are equal -- some are more important than others. IBM suggests using a scale of 1 to 100. When you run your tests, you can then use the weight to distribute your results. If a test **sort of** passes (some things don't fully work or don't work on certain configurations but do work on others, for example), you can say that 70% pass and 30% fail by using the weight sliders. With a weight of 1, this is impossible. See Figure 2 for an example of defining test case weigh.

The weight concept is an important feature in Rational Quality Manager. It moves beyond the shallow "pass" or "fail" qualifiers. It's unusual to run a test case or a charter (that isn't automated) where you can say that it passed 100%. In the past, you didn't have the option of saying "Well, it mostly works, but there are a couple of issues." Now, by specifying a test case weight, you can.
Assigning test case categories
Just like tags for requirements, categories of your test cases can be invaluable for slicing and dicing your testing data so that you can determine what's really happening. You have three default options: **Category**, **Function**, and **Theme**, and you can add, delete, or edit them. On a default installation, these drop-down menus are probably blank. However, there are several places where you can add values to them (as an Admin, or simply by clicking the "Manage Test Case Categories" icon, which is in both the test plan and test case).

As with requirements, you can use the **category** and **function** to specify where you are testing (reporting, core platform, voice, internal tools, external tools, or administration, for example) and **theme** to specify why you are testing (capability, performance, security, testability, supportability, or scalability, and so on). You can define them however you want, but think about how you might divide up your test effort and how you want to report on the results. Later, you can use these fields to compare and contrast coverage and results for different areas of the application and different types of testing. Figure 2 shows an example of using test case categories.

**Figure 2. Defining test case category, function, theme, and weight**

Use test planning tools to track progress
Reporting testing progress is sometimes a forensic analysis of testing results, and sometimes it's reporting on high-level milestones, state gates, or entry and exit criteria. That's where the test planning features of Rational Quality Manager come into play. It should come as no surprise that one of the key tools for tracking and reporting testing progress is the test plan. The default test plan template included in Rational Quality Manager provides several features that assist you both in knowing where the test project is and communicating it to others.

Tracking requirements
As described in detail earlier, Rational Quality Manager has several requirements features that help you manage requirements coverage. In the test plan, there is a Requirements section (see Figure
3) for managing all of the requirements that you will be covering in a given test plan. This is helpful because even if your application requirements are managed in a different tool and you merely import them into Rational Quality Manager, you can define *your own* test requirements in your plan.

**Figure 3. Example of requirements in a test plan**

It is not uncommon to see great functional requirements coverage (the application should do X, but it should not do Y), yet no requirements for parafunctional requirements. That does not mean that we do not test them. We do. However, it is always difficult to keep track of where that testing is in terms of status and coverage. By creating your own requirements, you can add requirements for performance, security, usability, and other areas that are often overlooked. You can then tie test cases back to those requirements to track coverage and status.

**Tracking test environments**

If your application has configuration testing of any type (support for multiple hardware platforms, operations systems, browsers, integration with other vendors or programs, or another configuration setting), take the time to set up test environments in Rational Quality Manager. In the Test Environments section of the test plan (shown in Figure 4), you can specify your actual test environments, as well as the platform coverage.
**Figure 4. Example of platform coverage in the test environments section of a test plan**

The test environment description shows the combination of hardware and software platforms to be used in testing. This is usually not an exhaustive list, because it is not always possible to cover everything. But again, when it is time to run your tests, if you have everything set up properly, it is easier to see which platforms or configurations you have or haven't tested. In Rational Quality Manager, especially when using the Rational Test Lab Manager add-on, the test environments are associated with actual test runs.

**Tracking quality objectives**

As you may already have picked up from earlier statements here, projects sometimes focus overwhelmingly on requirements coverage, and the majority (99% or more) of those requirements are related to functionality. That's a shame, because you can miss a lot in your testing because of that focus. IT can be a painful process to shift the testing into other areas. It is important to spend a lot of time testing functionality (or capability), but it is also important to spend a lot of time testing other quality criteria.

In Rational Quality Manager, you can explicitly define your quality objectives in the Quality Objectives section of the test plan (Figure 5). This section lists your quality objectives for a release, in table format. You can freely edit the Quality Objectives Description, the Current Value, and the Comment field (not shown here) to specify just about any objective.
These are just examples of measurement objectives:

- Code complexity
- Unit testing success
- Code coverage
- Requirements coverage
- Test case completion (percent complete, percent pass) by area
- Load, performance, or scalability
- Open issue and defect severity, volume, or status
- Defect arrival rates or testing velocity
- Test case or requirement priority or severity, or both
- Compliance with a standard (Section 508, W3C,)
- Documentation or evidence requirements

There are many, many more. The quality criteria that you choose will depend heavily on what you are trying to accomplish with the project and what development context you're working in, of course. Whatever you choose, the Quality Objectives section provides a snapshot of where the project is, from a quality management perspective.

**Tracking entry and exit criteria**

In the trend of Quality Objectives, you can also specify entry and exit criteria. In those sections of the test plan, you can elaborate on the criteria that will support your testing process, as well as your overall quality criteria. You can use entry criteria (shown in Figure 6) to specify the conditions required to begin testing, such as the minimum level of product and feature quality necessary.
Exit criteria (Figure 7) can be used to specify the conditions to meet for a particular test cycle to be considered complete. For example, you might specify that testing is incomplete until all of the most severe defects have been fixed.

**Analysis using viewlets and reports**

Moving past planning, goals, and communicating high-level goals and progress, you might be wondering how you can look at the real data in Rational Quality Manager. Well, in this section, you will see several of the reports and viewlets in Rational Quality Manager. For many of these, you can see them as both viewlets (those little windows that you can create on your dashboard to show real-time updates on status) and reports (configurable reports that find targeted data).

This is not a complete listing of the data that you can review; it's not even close. It is simply a glimpse of some of the analysis that you can start to do. All of these charts are from preformatted reports that are included in the software. In addition to the provided reports, you can use the data in Rational Quality Manager to create or support custom reports based on specific project needs and data gathered from other test management tools.

**Tracking test execution**

By default, Rational Quality Manager includes reporting on the status of test plan execution, on trends, and on defects. Examples of several of the reports that are available follow, along with brief descriptions. Execution status reports by plan, owner, and machine each display charts with data that is divided into six color-coded categories. All of these reports use the same status outcomes.

**Live execution status report**

The live execution status viewlet is one of the default viewlets that you find when you first log into Rational Quality Manager. It is configurable but by default it shows the status of test execution by test plan within your project. Multiple test plans can be shown side-by-side. There are several states available by default, and clicking on any given plan or state segment takes you to details for that selection. An example of a Live Execution Status viewlet is shown in Figure 8.
Execution status according to tester
This report lists the status of execution work items by their testers, or owners. You can select more than one plan to see the status of execution work items by owners across multiple plans. As with all of the other reports, you can click a section of the graph to view the execution work items that are associated with a particular status for that owner. An example of an Execution Status per Tester report is show in Figure 9.

Execution trend report
The Execution Trend report can be used for comparing the actual test execution progress to the projected progress. It compares what you did with what you planned to do. It also shows how
much work is left and how you’ll need to change velocity if you want to stay on target. This report gives you a good indication of velocity over time. Figure 10 shows an example of an Execution Trend report.

**Figure 10. Example of an Execution Trend report**

![Execution Trend report example](image)

**Tracking requirements coverage**

It probably comes as no surprise that you can use Rational Quality Manager to get a report of the status of requirements coverage. Although it is good to look at other measures, too, this can still be important. The following subsections explain the most useful basic reports.

**Requirements Coverage report**

Again provided as a default viewlet on your dashboard, this report shows your overall requirements coverage. The pie chart is divided into two sections: Covered and Not Covered. Click either section to get the details of the requirements and test cases. Figure 12 shows an example of a Requirement Coverage Status report.
Figure 11. Example of the Requirements Coverage Status viewlet

Clicking on **Covered** produces the table shown in Figure 12, which shows each requirement and its associated test cases.

Figure 12. Plan Requirements Coverage detail view

<table>
<thead>
<tr>
<th>Requirement ID</th>
<th>Requirement Name</th>
<th>Test Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>Require support for Feature6</td>
<td>Web Services Editors Test 1</td>
</tr>
<tr>
<td>10</td>
<td>Require support for Feature3</td>
<td>Reporting Editors Test 2</td>
</tr>
<tr>
<td>12</td>
<td>Require support for Feature5</td>
<td>Reporting Editors Test 2</td>
</tr>
<tr>
<td>9</td>
<td>Require support for Feature2</td>
<td>Rich Client UI Login Test 3</td>
</tr>
<tr>
<td>11</td>
<td>Require support for Feature4</td>
<td>Web Services Login Test 7</td>
</tr>
<tr>
<td>7</td>
<td>Require support for Feature1</td>
<td>Reporting Dashboard Test 6</td>
</tr>
<tr>
<td>6</td>
<td>Require support for Feature1</td>
<td>Reporting Dashboard Test 6</td>
</tr>
</tbody>
</table>

Clicking on **Not Covered** brings up the table shown in Figure 13, which shows each requirement not covered and who owns that requirement. (Perhaps the rationale for this is that, by knowing who the owner is, you can gently prod that person to write a test case.) You can also drill down in that information to break out coverage by test plan.
Figure 13. Table for detailed review of requirements not covered

<table>
<thead>
<tr>
<th>Requirement ID</th>
<th>Requirement name</th>
<th>Requirement owner</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Now Customer Order Requirement</td>
<td>mary</td>
</tr>
</tbody>
</table>

Requirements Status by Execution report
The Requirements Status by Execution report shows the status of the work items for each requirement in a test plan. You can view this report using Count or Weight. Figure 14 shows an example.

Figure 14. Example of a Requirements Status by Execution report

Tracking test cases
If you are a test manager who likes to look at test ideas to understand what testers are focused on, you will like how easy it is to get details about test cases in Rational Quality Manager. You don’t get just a huge list of test cases. Instead, you get lists that you can sort and filter by using some of the categories and tags mentioned previously. You can use the default test case reports to list test cases by plan, configuration, or team.

Test Cases by Plan report
The Test Cases by Plan report queries all test cases that are part of a test plan. You can click the name of the test case to view the test case. See the example in Figure 15.

Figure 15. Example of a Test Case by Plan report

<table>
<thead>
<tr>
<th>Test case name</th>
<th>Last modified</th>
<th>Category</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reporting Dashboard Test 6</td>
<td>Jun 20, 2008 9:23 AM</td>
<td>Reporting</td>
<td>Dashboard</td>
</tr>
<tr>
<td>Reporting Editors Test 2</td>
<td>Jun 20, 2008 9:23 AM</td>
<td>Reporting</td>
<td>Editors</td>
</tr>
<tr>
<td>Rich Client UI Execution Test 6</td>
<td>Jun 20, 2008 9:23 AM</td>
<td>Rich Client UI</td>
<td>Execution</td>
</tr>
<tr>
<td>Rich Client UI Login Test 3</td>
<td>Jun 20, 2008 9:23 AM</td>
<td>Rich Client UI</td>
<td>Login</td>
</tr>
<tr>
<td>Web Services Admin Test 5</td>
<td>Jun 20, 2008 9:23 AM</td>
<td>Web Services</td>
<td>Admin</td>
</tr>
<tr>
<td>Web Services Dashboard Test 4</td>
<td>Jun 20, 2008 9:23 AM</td>
<td>Web Services</td>
<td>Dashboard</td>
</tr>
<tr>
<td>Web Services Editors Test 1</td>
<td>Jun 26, 2008 11:05 AM</td>
<td>Web Services</td>
<td>Editors</td>
</tr>
<tr>
<td>Web Services Login Test 7</td>
<td>Jun 20, 2008 9:23 AM</td>
<td>Web Services</td>
<td>Login</td>
</tr>
</tbody>
</table>
There is a very similar report for Test Cases by Team, which looks and works the same. That report looks across teams and the test cases assigned to the people in those teams.

**Test Cases by Configuration report**

Similar to the Test Case by Plan report but for those who do configuration testing, the Test Cases by Configuration report sorts test cases by target configuration. Figure 16 gives you an idea of its usefulness.

**Figure 16. Example of a Test Case by Configuration report**

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Test case name</th>
<th>Category</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAMPLE AMD x64 Linux Firefox</td>
<td>Reporters Editors Test 2</td>
<td>Reporting</td>
<td>Editors</td>
</tr>
<tr>
<td>SAMPLE AMD x86 WinXP Firefox</td>
<td>Web Services Admin Test 5</td>
<td>Web Services</td>
<td>Admin</td>
</tr>
<tr>
<td>SAMPLE AMD x86 WinXP IE</td>
<td>Web Services Login Test 7</td>
<td>Web Services</td>
<td>Login</td>
</tr>
<tr>
<td>SAMPLE Intel x86 WinXP IE</td>
<td>Web Services Dashboard Test 4</td>
<td>Web Services</td>
<td>Dashboard</td>
</tr>
<tr>
<td>SAMPLE x86 Linux Firefox</td>
<td>Rich Client UI Login Test 3</td>
<td>Rich Client UI</td>
<td>Login</td>
</tr>
<tr>
<td>SAMPLE x86 WinXP IE</td>
<td>Web Services Editors Test 1</td>
<td>Web Services</td>
<td>Editors</td>
</tr>
<tr>
<td></td>
<td>Reporters Dashboard Test 6</td>
<td>Reporting</td>
<td>Dashboard</td>
</tr>
<tr>
<td></td>
<td>Rich Client UI Execution Test 8</td>
<td>Rich Client UI</td>
<td>Exec</td>
</tr>
</tbody>
</table>

**Next steps**

That gives you an overview of what some of your options are for test analysis and reporting in Rational Quality Manager. The next step is to try it for your project. As you work with the data, see if you can think of changes that you could make at the outset to make reporting easier later. Also, think about other types of information that you are not getting from the default reports. The next step would be to consider creating your own by using the Create Report feature.
Related topics

- For more on getting started with Rational Quality Manager, check out *Managing your first project with IBM Rational Quality Manager* by Michael Kelly (IBM developerWorks, October 2008).
- Download a trial version of *Rational Quality Manager*.
- To find more information about IBM Rational Quality Manager and related products, check the [Software testing and quality management](https://www.ibm.com) page on IBM.com.
- You can find more information about IBM Rational Quality Manager and related products on the [Software Testing and Quality Management](https://www.ibm.com) of IBM.com.
- Subscribe to an [IBM developerWorks newsletter](https://www.ibm.com/developerworks).
- Try [IBM Bluemix](https://www.ibm.com) free for 90 days. Bluemix is a cloud platform as a service (PaaS). Bluemix supports several programming languages and services as well as integrated DevOps to build, run, deploy and manage applications on the cloud.

© Copyright IBM Corporation 2008
**Trademarks**
(www.ibm.com/developerworks/ibm/trademarks/)