Effective management through measurement

Doug Ishigaki

May 14, 2004

From The Rational Edge: Using sample views generated by automated tools in the IBM Software Development Platform, Ishigaki shows how an automated measurement program can help software project managers assess progress, mitigate risks, and improve team productivity.

Managing software development is difficult for project managers with any amount of experience, and obtaining accurate measures of project progress is a constant challenge. Typically, when project managers rely on team members to provide status data, they get results that are inconsistent, imprecise, and hard to collate. In addition, throughout the project, executives, functional managers, and clients may ask for status reports containing selected data in various formats. Too often, teams find they are spending more time reporting on status than on developing the solution.

An efficient, automated software project measurement program can help project managers cope with these challenges. It enables them to assess progress in multiple project areas, identify problems early in the development lifecycle, and improve team productivity by eliminating tedious reporting tasks.

In this article, we will discuss the benefits of using such a measurement program and identify best practices and tools that can support it. Throughout the article, we will examine sample measurement views for both executives and project managers, generated by automated tools in the IBM Software Development Platform.
Measurement as an effective management tool

Measurement is an effective management tool because it provides the information decision makers require to accurately monitor key issues related to company and organizational goals, progress and quality at both the executive and project levels, and performance against a plan. Measurement also provides data that managers need to ask the right questions -- and make the right decisions based on objective information.

The January 2003 *Rational Edge* article "Practical Measurement in the Rational Unified Process" detailed several benefits that bear repeating here, along with new advantages we have identified. As managers, measurement helps us do the following:

- **Communicate effectively and improve visibility.** Measurement supports communication among stakeholders across all levels of the organization. Objective measurement also reduces ambiguity. It provides an effective way to communicate status between suppliers and their clients.

- **Identify and correct problems early.** Measurement facilitates pro-active management. It allows us to identify and manage potential problems early in the development lifecycle. Problems discovered late are more difficult to manage and more costly to fix. With measurements, managers do not have to wait for problems to arise; instead, they can anticipate and quickly address them.

- **Make key trade-offs.** Decisions in one area often impact others. Measurement helps assess the impact objectively so that managers can make informed trade-off decisions to best meet project objectives.

- **Track specific project objectives.** Measurement helps managers answer specific questions such as: "Is the project on schedule?" or "Is the quality improving?" or "Is the system ready to be delivered?" By tracking actual measures against a plan, managers can assess progress toward project and organizational objectives.

- **Manage risks.** Risk management is a widely accepted best practice that involves identifying and analyzing risks early in a project's lifecycle. Risks uncovered late can be difficult and costly to fix. With high-quality objective data, managers can gain visibility into risk areas such as requirements creep. By measuring and monitoring requirements volatility, they can determine whether a risk has been mitigated.

- **Defend and justify decisions.** Managers must effectively defend and justify their decisions. Given a choice between basing their decisions on subjective or objective data, the vast majority would choose objective data. Measurement provides objective historical performance or trend data as well as current performance data. It also provides perspective with respect to time, projects, releases, and so forth. This allows decision makers to interpret the measures and decide on appropriate actions.

- **Plan future projects.** In project planning, managers must set realistic goals and schedules, not to mention budgets. Recording timelines and expenditures for past projects provides the data that managers need to predict schedules and costs for similar projects.
Measurement and process

It has been proven many times: Companies and teams that develop software successfully follow a repeatable and measurable process. At IBM Rational, we have also discovered how these companies and teams relate process to measurement. Successful organizations do the following:

- Establish the underlying process first.
- Use measures that are natural by-products of the underlying process.
- Follow the underlying process to make the measures valid.
- Improve the quality of their measures via automation, which produces more objective results.
- Identify instrumentation points in the process that they can translate into measures or trackable measurement attributes.

In working with many organizations, we have also identified common problems relating to measurement and process and helped managers find solutions.

- In some organizations the software development process is rigid, and measurement is secondary to this process. If you need a new measurement, you first need to update or change the process. You can resolve this problem by tailoring the process and incorporating instrumentation points directly into it. You can make these points easily accessible by deploying development tools to do the instrumentation (see Figure 1). Automated tools make it much easier to define what you can and will collect.
- Sometimes measurements do not reflect actual work. This is a problem if the organization can (or chooses to) track only project budget and schedules. In such cases, the amount of money or time spent on a task may not reflect the actual work performed. For example, you may know you spent X amount of dollars developing code, but you won’t know how much code was actually developed. You cannot measure productivity by tracking only task schedules and expenditures. Again, by modifying your process to include additional measurements of project artifacts, such as requirements, use cases, design objects, code, test cases, defects, and so forth, you can get a more accurate picture of what your time and money actually bought and how far you have progressed.

Using automated tools like those shown in Figure 1 can help make measurement a key component of the software process and provide easy access to instrumentation points.

Figure 1: Tools in IBM Software Development Platform that are useful for measurement

Key management measures and views

How should you modify your process and what should you measure? We are often asked these questions. And often, our answer is, "It depends." First, it depends on where you are in the software development lifecycle. If you're doing iterative development, what you want to measure in the Elaboration phase will be different from what you measure in the Construction phase. It also depends on your position in your organization. If you're a division manager, you'll want different information than a project manager. You may be interested in improving productivity, whereas a project manager will be interested in meeting the proposed schedule and reducing the number of product defects.
The key is always to know your information needs and then design your measures. You can define these needs by organization, project, team objectives, issues, and risks. Furthermore, you'll want to categorize your measures in order to make them understandable to the intended users. As an example, in this article we will organize management measures based on IBM Software Development Platform capabilities, shown in Figure 1.

IBM Software Development Platform

A comprehensive set of tools, proven best practices, and professional services, IBM Software Development Platform helps teams build, integrate, extend, modernize, and deploy software and software-based systems. Using IBM Rational Unified Process®, or RUP®, as a foundation, the platform automates and integrates the software development lifecycle using an iterative development approach. Platform tools enable companies to focus on architecture, continually ensure quality, and manage change and assets so that they can better leverage their software development capability.

IBM Software Development Platform spans all areas of software development:

- **Process and project management.** Managers can plan and implement measures for every project and ensure consistent processes across the enterprise.
- **Requirements and analysis.** Automated tools help reduce risk by enabling managers to understand and manage the evolution of business and software requirements throughout the software development lifecycle.
- **Design and construction.** Teams can maximize their productivity while building business applications, software products, and systems.
- **Software quality.** Teams can increase predictability and ensure quality by uncovering defects early in the development lifecycle, when they are easier to address and less expensive to fix.
- **Software configuration management.** The platform includes tools for securely managing software changes and assets, providing traceability throughout the development lifecycle.

Let's look at examples of management measures you can implement with IBM Software Development Platform for each of these areas. These examples were generated using IBM Rational® ProjectConsole, a component of IBM Rational Team Unifying Platform, which is itself a component of IBM Software Development Platform.

Rational ProjectConsole automates reporting on project status, providing a number of automated collection agents, customizable reporting, and a graphical dashboard, which allows you to customize measurement views and perform drill-down, root-cause analysis.

Measuring project management and process

A successful measurement system must be flexible enough to satisfy the information needs of an organization. Measurement requirements vary from high-level views for executives at the C-level (CTO, CIO, CEO, etc.) or division manager level, to more detailed views at the project manager level.

Executives usually need high-level information. For example, the division manager responsible for a number of projects or products would like to determine, at a glance, the status of each
project and key progress measures. A green-yellow-red visual status indicator can provide this high-level shorthand. The example view in Figure 2 displays information generated by Rational ProjectConsole for a list of projects. It includes:

- **Status**: Overall project status based on a calculated value compared to a known threshold. The actual calculation is organization- or project-specific.
- **Open High-Priority Change Requests**: The number of high-priority change requests that are currently open. In this example, this measurement was identified as an information need.
- **Glidepath**: The number of defects that have been verified versus the number of forecasted defects to be verified. In planning a release, the projects identified n-number of defects that teams planned to fix in the release. This measurement shows progress toward the goal.
- **Test Coverage**: Percent of the code that has been tested.
- **Change Requests to Verify**: How much work is left to perform.

![Figure 2: Executive view for a list of projects](image)

Figure 2 shows another executive view with a visual (green-yellow-red) status indicator of project health. The executive can also drill down on the project name to view a second level of detailed status information (Figure 4) for each project.

![Figure 3: Executive view of a division status report](image)

Figure 3: Executive view of a division status report

At the project level, typical areas of interest for overall status reporting are:
- **Project management**: Cost, schedule, risk, staffing, and so forth.
- **Process**: Compliance, conformance, productivity goals, and so forth.
- **Requirements and analysis**: Progress in meeting requirements, volatility, and use-case development.
- **Design and construction**: Completion of, and changes to, business, design, and database models; code development.
- **Software quality**: Unit test progress, system test progress, test coverage.
- **Software configuration management**: Defect and change request tracking, baseline progress, activity management.
- **Engineering support**: Tool development, software development environment support.
- **Operations**: Hardware availability.

Figure 4 shows an example project manager view.

![Figure 4: Project manager's high-level status report for all areas](image)

A manager might choose to drill down into the areas shown in Figure 5 to gather more status information related to the following project management categories:

- **Highlights and issues**: To identify, analyze, and track action items and issues.
- **Planning and schedules**: To monitor the project plan and schedules and track progress (work completed over time) against the plan. One useful schedule indicator is the Schedule Performance Index (SPI), shown in Figure 6. SPI is a calculated index. If the SPI is greater than one, the project is ahead of schedule. If it is less than one, the project is behind schedule.
- **Cost and earned value**: To maintain control over expenditures throughout the project lifecycle. A useful cost indicator is the Cost Performance Index (CPI) shown in Figure 6. Like
the SPI, the CPI is calculated. If the CPI is greater than one, the project is over budget. If it is less than one, the project is under budget.

- **Staffing**: To monitor project member additions and attrition as indicators of project momentum and morale. Sharp increases in staff can slow project progress as new people are brought up to speed. A low attrition rate for good team members is a sign of success. High turnover may indicate problems in morale or motivation; it is a warning sign for project managers.

- **Risk**: To continually identify, analyze, and track risks that threaten project success. Pro-active risk management increases the chance of project success.

- **Compliance**: To track compliance with standards or process improvement goals.

- **Customer satisfaction**: To ensure that customer needs are satisfied.

Project managers may pick and choose what categories to monitor on this list.

<table>
<thead>
<tr>
<th>Project Area</th>
<th>Category</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Management</td>
<td>General</td>
<td></td>
</tr>
<tr>
<td>Project Management</td>
<td>Highlights and Issues</td>
<td></td>
</tr>
<tr>
<td>Project Management</td>
<td>Planning</td>
<td></td>
</tr>
<tr>
<td>Project Management</td>
<td>Staffing</td>
<td></td>
</tr>
<tr>
<td>Project Management</td>
<td>Earned Value</td>
<td></td>
</tr>
<tr>
<td>Project Management</td>
<td>Schedule</td>
<td></td>
</tr>
<tr>
<td>Project Management</td>
<td>Risk</td>
<td></td>
</tr>
<tr>
<td>Project Management</td>
<td>Compliance</td>
<td></td>
</tr>
<tr>
<td>Project Management</td>
<td>Customer Satisfaction</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 5: Project management status report**

**Figure 6: Cost Performance Index (CPI) and Schedule Performance Index (SPI)**

Figure 7 shows progress against the plan. Automated project plans let you identify tasks and assign scheduled completion dates. Using IBM Rational ProjectConsole, you can track the progress of tasks in your project plan as you work toward completion.
Figure 7: Task progress -- planned tasks versus tasks complete

In the process area, a project manager may want further status information about the following:

- **Highlights and issues**: To identify, analyze, and track action items and issues.
- **Planning and schedules**: To monitor the project plan and schedules for rolling out the process.
- **Cost and earned value**: To track expenditures and their results.
- **Staffing**: To track project member additions and attrition.
- **Risk**: To identify, analyze, and track risks and mitigation efforts.
- **Compliance**: To determine compliance with standards or process maturity goals.
- **Defects contained**: To monitor defects found and fixed during a specific time period or release. This measure may be an indicator of test effectiveness.
- **Defects escaping**: To measure the number of post-release defects. This measure may indicate problems in the software development process.
- **Scrap and rework**: To monitor the amount of change to the software baseline. In a healthy project, this trend is decreasing or stable. Increasing rework is a sign of increases in maintenance demands and costs. Figure 9 shows scrap in line of code. However, scrap and rework are not limited to code; they also result from design changes and can be monitored in the design model.
- **Productivity**: To determine whether the underlying process is improving productivity.

Figure 8 shows a high-level green-yellow-red status report for process.
Measuring requirements and analysis

Requirements and analysis involves understanding the problem space, capturing and managing evolving requirements, modeling user interactions, and incorporating stakeholder feedback throughout your project lifecycle. Good requirements and analysis practices help reduce project risks.

Areas to monitor and measure within requirements and analysis include:

- **Highlights and issues**: Action items and risk mitigation in requirements and analysis.
- **Planning and schedules**: Plan versus actual requirements or use cases developed against schedules.
- **Cost and earned value**: Budget tracking for requirements and analysis.
- **Requirement reviews**: Progress on requirements and stakeholder reviews.
• **Use-case reviews**: Progress on use cases and stakeholder reviews.
• **Inspections**: Requirements verification.
• **Requirements volatility**: Changing requirements impact schedules and cost. See Figure 13 for a detailed look at this measure.

A high-level view for a project manager could show green-yellow-red status indicators for each area, as shown in Figure 10.

<table>
<thead>
<tr>
<th>Requirements &amp; Analysis</th>
<th>Highlights and Issues</th>
<th>●</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements &amp; Analysis</td>
<td>Planning</td>
<td>●</td>
</tr>
<tr>
<td>Requirements &amp; Analysis</td>
<td>Staffing</td>
<td>●</td>
</tr>
<tr>
<td>Requirements &amp; Analysis</td>
<td>Earned Value</td>
<td>●</td>
</tr>
<tr>
<td>Requirements &amp; Analysis</td>
<td>Use Case Model</td>
<td>●</td>
</tr>
<tr>
<td>Requirements &amp; Analysis</td>
<td>Requirement Reviews</td>
<td>●</td>
</tr>
<tr>
<td>Requirements &amp; Analysis</td>
<td>Use Case Reviews</td>
<td>●</td>
</tr>
<tr>
<td>Requirements &amp; Analysis</td>
<td>System Inspections</td>
<td>●</td>
</tr>
<tr>
<td>Requirements &amp; Analysis</td>
<td>Requirements Volatility</td>
<td>●</td>
</tr>
</tbody>
</table>

**Figure 10: High-level status view for requirements and analysis**

Additional detailed views, such as the requirements summary in Figure 11 and the use-case summary in Figure 12 (both extracted from IBM Rational RequisitePro), can provide specifics about requirements and analysis development progress.
Figure 11: Requirements summary view

Figure 12: Use-case summary view

Figure 13: Requirements churn (volatility) -- measured by number of revisions

Measuring design and construction

Design and construction involves architecting and modeling for design, model-driven development, software coding, fixing defects and responding to enhancement requests, and component testing.

Within design and construction, areas to monitor include:

- **Highlights and issues**: Action items and risk mitigation.
• **Planning and schedules**: Planned design elements or source code versus actuals and against schedules.
• **Staffing**: Additions and attrition among development staff. This measure is important on large projects.
• **Cost and earned value**: Expenditures and budget tracking.
• **Design model completeness**: Completeness of your architecture and design model, or design model progress (trend). See Figure 15. Ideally, compare against planned values (not shown).
• **Size at complete**: Code development trends that show progress (see Figure 16). Ideally, compare actual code developed against planned values (not shown) to determine how much work remains to be done.
• **Software inspections**: Peer review inspections to uncover defects, inconsistencies, and trouble spots in products that might affect quality and performance.
• **Requirements volatility**: Changing requirements that impact delivery schedules and increase software rework and cost.
• **Open problems and originating dates**: Number of open problem reports, together with their age, that help determine whether staffing is adequate.

Figure 14 shows a high-level, green-yellow-red status view of design and construction for a project manager.
Figure 14: Design and construction high-level status view

Figure 15: Design model completeness measured by classes developed

Figure 16: Trend of total source lines of code (SLOC) developed

Measuring software quality

Ensuring software quality involves testing functionality, reliability, and performance. It also involves tracking all your test efforts and ensuring test coverage for all required functionality in an application targeted for release.
For software quality, areas to monitor include:

- **Highlights and issues**: Action items, issues, and risk mitigation.
- **Planning and schedules**: Planned test cases versus actual results and against schedules.
- **Staffing**: Additions and attrition among the test team.
- **Cost and earned value**: Budget tracking.
- **Unit testing**: Number of tests planned, executed, passed, and failed to determine progress.
- **System testing**: Number of tests planned, executed, passed, and failed (Figures 18 and 19).
- **Performance testing**: Throughput or response times.
- **Defects**: Tracking for customer-reported defects (Figure 20) or all defects (Figure 21).

For software quality, a project manager might view a high-level green-yellow-red status report like the one in Figure 17.

![Figure 17: High-level status view for software quality](image-url)
Figure 18: Test status (trend chart)

Figure 19: Test status by iteration
Software configuration management involves capturing, controlling, and managing software changes and assets. This includes defect and change tracking along with software asset management.

Within software configuration management, areas to monitor include:

- **Highlights and issues**: Action items and risk mitigation, which are usually necessary within all areas of a project.
- **Planning and schedules**: Source code planned for development versus actuals and against schedules.
- **Staffing**: Additions and attrition among configuration management staff.
- **Cost and earned value**: Budget tracking.
- **Defect status**: Targeted defect status.
- **Enhancement status**: Status of features under development.
- **Build status**: Build readiness.
- **Code churn**: Number of lines added, modified, and deleted in the configured or baselined software (Figure 23). This measure shows build and release readiness. Increasing churn indicates that the release is "not ready."

Figure 22 shows a high-level, project manager's view of the status for these software configuration areas.
Effective management through measurement

Summary

For managers, measurement provides progress assessment, insight into the quality of an evolving project, and data for objective decision making. Although it cannot guarantee a project's success, measurement does help decision makers take a proactive approach to managing critical issues inherent in software projects.

We have provided many examples of measurement views, but the key to success is not to implement all the measurements shown here. Rather, assess your project and organization, identify your stakeholders' needs, identify measures and indicators to ensure that you satisfy those needs, and implement a non-intrusive, automated measurement program that will provide the objective information you need to make key management decisions.

For more information about IBM Software Development Platform, see the IBM Web site. For information about IBM Rational ProjectConsole, see the IBM Team Unifying Platform page.

References

Learn more about Rational Performance Measurement and Management.


© Copyright IBM Corporation 2004

**Trademarks**