Software capability improvement.
White paper
May 2009

Software and system development with the IBM Measured Capability Improvement Framework.

A business process for smarter companies and smarter devices

Per Kroll
Chief Solutions Architect, Measured Capability Improvement Framework

and

Murray Cantor, PhD
IBM Distinguished Engineer, Rational CTO Office

Contents

Introduction

The pressure to improve the software and system delivery process has increased with the recent economic compression: Enterprises need to know that they are receiving sufficient value from the billions spent on software and systems delivery (SSD) each year. However, few organizations have sufficient governance of their SSD organization to ensure the desired overall SSD return on investment (ROI). That is because most organizations do not take the needed measurements to improve the business values and operational effectiveness of their SSD function. Even more organizations do not know what actions to take when measurements show unfavorable values: Most companies simply do not have a framework for knowing what changes to make and what to target for improvement.

Meanwhile, the need for more capable SSD organizations is growing. Our world is becoming smarter. In the workplace, we leverage software to collaborate, innovate, and make better decisions. All of us leverage the Internet to socialize and to gain information. Modern devices such as cell phones, automobiles, and even refrigerators are becoming increasingly instrumented, interconnected, and intelligent. Consider the Amazon's Kindle. This device consists of state-of-the-art hardware and embedded software, is connected via a cellular network to an Internet-based marketing portal that allows for device content management, handles purchases by connecting to financial services, and provides a channel for content providers. The Kindle requires an elegant integration of a wide range of software and systems.

The transformation to a smarter world relies on the ability to deliver software and systems, an increasingly important business process for virtually all enterprises. There is ongoing pressure to improve the business process of software and system delivery as it is becoming increasingly complex, with a need to support a multitude of models, such as packaged application development, outsourcing arrangements, distributed development, and enterprise modernization.

Unlike most other business processes, such as supply chain management or manufacturing, SSD needs to deal with a range of risk. SSD also differs from many other business processes in that it entails a diseconomy of scale: that is, individual productivity decreases with the size of the SSD effort.

In this paper, we describe how to govern the software and system delivery function to ensure favorable ROI by introducing a control framework that enables reasoning about those practices that work and those that don't. Using this framework permits us to effectively manage risk or innovation and diseconomy of scale, and to understand the effectiveness of difference development models.

Addressing risk and diseconomy of scale

Software delivery differs from many other business processes by dealing with a broad range of innovation. Some software projects, such as maintenance of existing systems, are reasonably predictable, similar to manufacturing processes. Those projects carry limited innovation and drive limited or no business differentiation. Other projects, such as building unprecedented and large software systems, require high degrees of innovation in addressing problems that have never been solved before on a schedule. Committing to delivering innovation requires assuming risk, since the lack of complete knowledge at project inception is inevitable and uncertainty regarding how to proceed is part of the challenge. This risk is manifested in the statistical variance in the estimate of the time or cost to complete.

A commitment to assuming risk entailed by bringing innovation to the enterprise provides the opportunity to improve ROI.

Another major difference between the business process of software delivery and other business processes is the diseconomy of scale. Typically, manufacturing and service delivery processes offer economy of scale: The cost of a unit of software grows nonlinearly (i.e., yields cost reduction) with the size and complexity of the system. But this is not the norm in software production.

To explain this, we can use a context map for understanding two dimensions, as shown in Figure 1:

- Risk / variance: The amount of uncertainty and hence risk associated with project parameters
- Scale: The size or complexity of a project.

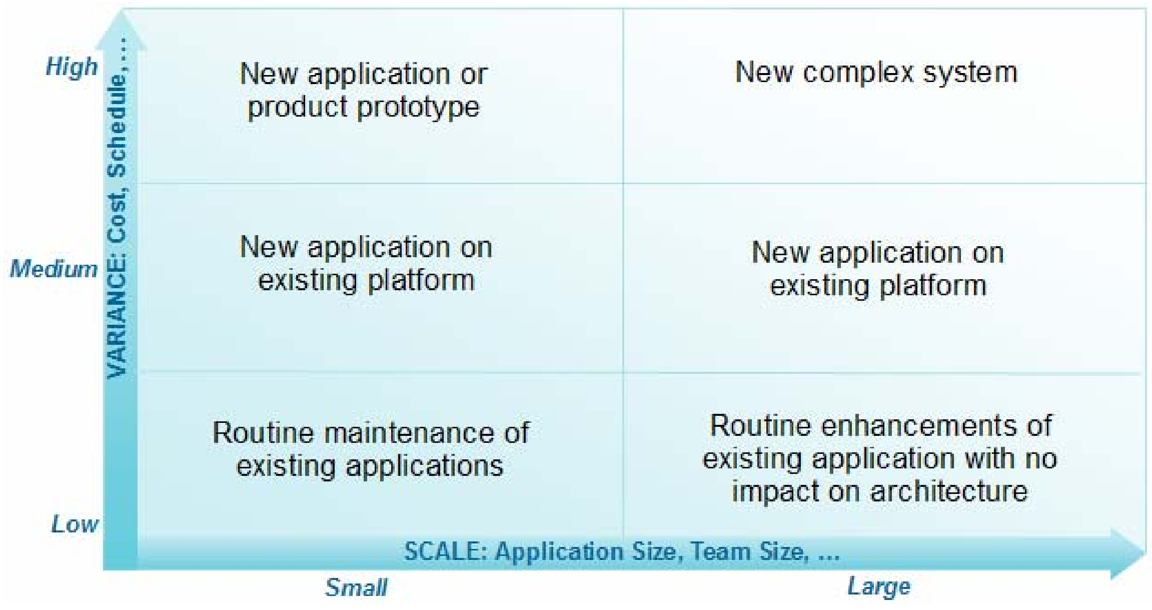

High
Medium
Low
VARIANCE: Cost, Schedule, ...
New application or product prototype
New complex system
New application on existing platform
New application on existing platform
Routine maintenance of existing applications
Routine enhancements of existing application with no impact on architecture
SCALE: Application Size, Team Size, ...
Small
Large

Figure 1. A two-dimensional context map facilitates discussions on which development strategies are appropriate for projects based on scale and risk/variance faced. The figure showcases a sample project type for each context.

We use the context map to analyze what strategies to apply for SSD improvement. As an example, many IT organizations spend the majority of their funds “keeping lights on” at the bottom third of this context map, versus providing new and business differentiating capabilities. The result is that the business views IT as a cost center versus a creator of business value. Let’s consider that same result in terms of financial investment strategies. Do you want to invest all your financial assets in low-risk instruments such as bonds? Low risk typically means low but safe return on investment. Such a strategy would be inappropriate if you seek high returns. Most financial planners suggest a healthy mix of low and high risk investments depending on your needs; software and system development should be no different. Using the context maps, one can determine appropriate strategies for an organization to invest in a more healthy mix of low-variance and high-variance projects. As an example, many organizations can drive major cost efficiencies in their low-variance projects through automation of change, build, and test environments, allowing them to invest the cost reductions into higher variance projects that have a higher return.

The context map also provides guidance on the set of practices SSD programs or projects should adopt. Projects in the upper right corner face both diseconomy of scale and high risk. Such projects need to excel not only in those practices that are essential in lower left -- such as change, build, and test automation -- but also in those practices related to:

- Agile governance – A means of enabling top business and technical risks to be attacked early on. Agile governance entails establishing well-defined business decision points along with feedback and direction to the team on where to respond with flexibility to new information to drive the program to success.
- Early architecture – Baselining of an executable (implemented and tested) architecture early in the program, combined with organizing around the architecture. This for example allows a 100-person development team to be divided into 10 smaller component teams that work semi-independently using a much more favorable software economics model (thus partially avoiding the diseconomy of scale).
- Agile development – Each of the component teams mentioned above can now adopt agile development techniques, improving their efficiency and ability to adapt to changing business and technical requirements.

A control framework for software delivery

Over the last two decades, the IBM Rational organization has helped thousands of businesses improve the way they deliver software and systems. We have learned that certain approaches to software and systems delivery drive more successful outcomes. In particular, SSD can be managed as a business process if there is a control framework that accounts for the different areas in the context map. To drive the appropriate changes, and measure and respond to the outcome for continual improvement, the control framework must be tailored to address the following concerns:

- How do I minimize the amount of change I should take on, while maximizing the return?
- How do I know that my organization is effectively implementing the required change?
- How do I know that the change is leading to improved operational or business results?

The IBM® Rational® Measured Capability Improvement Framework (MCIF) described in the following section provides the means to address these concerns.

The MCIF

To address these concerns and generally enable SSD improvement, we adapted the common three-tiered process improvement framework to account for the range of context described above (see Figure 2.) The upper tier looks at the measures of what the organization provides the broader enterprise. The middle tier measures the operational efficiency or effectiveness of the organization as it meets the business measures. The bottom tier is a measure of the internal controls the organization uses to improve the upper tier measures.

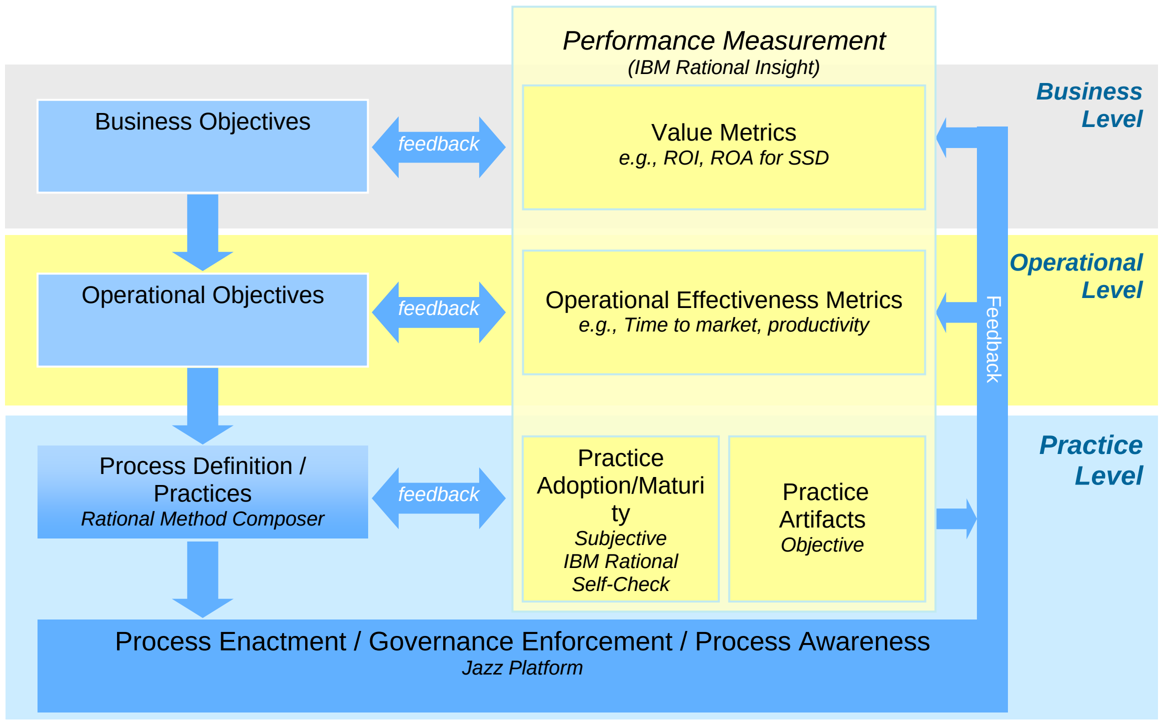


Figure 2: MCIF provides a Control Framework for Software Delivery.

One cannot control SSD efforts or organizations like a factory. SSD is more like an economy.[1] Even so, there are control mechanisms. MCIF is based on a key principle: The primary way to effect improvement in an SSD organization is to select and mature practices based on the SSD context. In effect, one controls the SSD organization by way of the practices.

Note that MCIF has three levels:

- Business value – the return the SSD organization returns to the business for the monies spent (an outcome).
- Operational effectiveness – the measures the SSD organization uses to ensure it does deliver value (an outcome).
- The practices and the measurement of their adherence and maturity (the controls).

The framework is completed by providing the feedback mechanisms that show the impact of the controls on the outcomes.

To apply the MCIF, teams first identify target business objectives and the operational goals that, if met, will enable meeting the business objectives. Then they determine which IBM® Rational® Method Composer practices to focus on based on the context and operational goals, and then institutionalize the practices through enablement and automation. Finally, they put in place the business analytic and statistical process control framework to collect the measures and determine the impact of the controls on the outcomes.

To achieve the business and operational goals, you should first measure how well the adoption is going through subjective and objective control metrics. Take corrective actions as needed, such as additional focus on problematic practices or organizational units, and take a step back every few months to see if the adoption of practices and associated technology has allowed the organization to reach target operational and business objectives. Apply the process control framework to determine which adoption practices need further focus or decide to extend the set of practices.

Applying MCIF

No technical changes should be done without a link to the expected business value. We do not improve quality for the sake of improving quality, but to allow our business to become more competitive. Applying MCIF gives the organization the measures needed to understand what works and what does not. It also provides a framework enabling discussions around the business value of IT.

Outcome Metrics

Outcome measurements address the set of operational and business objectives. Examples of Operational Objectives include Reduce Time-to-Market, Reduce Cost / Increase Productivity, Improve Quality, Increase Predictability, and Satisfy Compliance Mandate. These are important goals, and should make a difference in business operations. But how do you measure them? We typically need to drill deeper into what the target organization intends in assigning these goals. The following are some examples of drill-downs, including how neglecting to analyze the intention behind the objectives can lead to wrong behaviors:

- Reduce Time-to-Market: Should the start time be when you get an idea, or when you formalize a project? Should the end time be when development and testing has been completed or when the application has been successfully adopted? What matters to your organization? A few years back, the IBM Rational organization used “development complete” as the end date, which led development teams to cut corners in enablement and deployment assets, increasing rather than decreasing the time before customers saw value in our products.
- Predictability: Most teams struggle with predicting the likelihood of a project to release on time. IBM Research has developed technology based on the insight that the time-to-complete is at best an estimate that must be specified as a statistical distribution. The uncertainty of delivery then can be measured and managed by treating the current variance in the estimate as a project parameter. The IBM Rational organization will be embedding this capability in its products.

These are only some examples, but they highlight a need for deeper analysis of what the adopting organization is trying to achieve to determine the right measures.

Example Business Objectives include organization profitability, profit per employee, market-share growth, return on investment, and return on assets. The key is to establish how IT contributes to the business to move IT from a cost center to a value creation center.

Many organizations struggle with how to assess the business value of their project portfolio. IBM Research working with the Rational brand team has built a tool that enables organizations to measure and manage the investment value of their portfolio of ongoing development and delivery efforts. This tool will be described more fully in a separate article.[2]

We will see in later sections examples of some of the metrics used within IBM Software Group.

Control Metrics

Especially in today's climate, we do not have the luxury of trying something out, and finding out 18 months later whether it worked or not. We need rapid feedback on what works and what does not so we can take corrective action. Using a factory metaphor, you can see the adoption of practices described in previous section as the knobs in the software delivery factory that allows us to tune the effectiveness of the factory. We now need to know whether our software delivery factory is properly responding to the turning of the knobs. For this, we need so called **Control Metrics** which allow us to understand whether we are getting better at Iterative Development, Continuous Integration, and Use-Case-Driven Development.

IBM® Rational® Self-Check for Software Teams has been used within IBM since 2002. Self-Check provides teams with a simple mechanism for establishing a set of Control Metrics by assessing themselves, enabling the team to agree on a set of change actions to react to unfavorable control metrics, that is, problem areas in their practice adoption, as shown in Figure 3. One of the strengths with Self-Check is that it engages the team in the change effort, which is crucial to effectively driving organizational change. One of the drawbacks is that the Control Metrics are subjective, making them counter-productive to use for score carding, since score carding would encourage team members to alter their subjective responses to game the system to a favorable score.

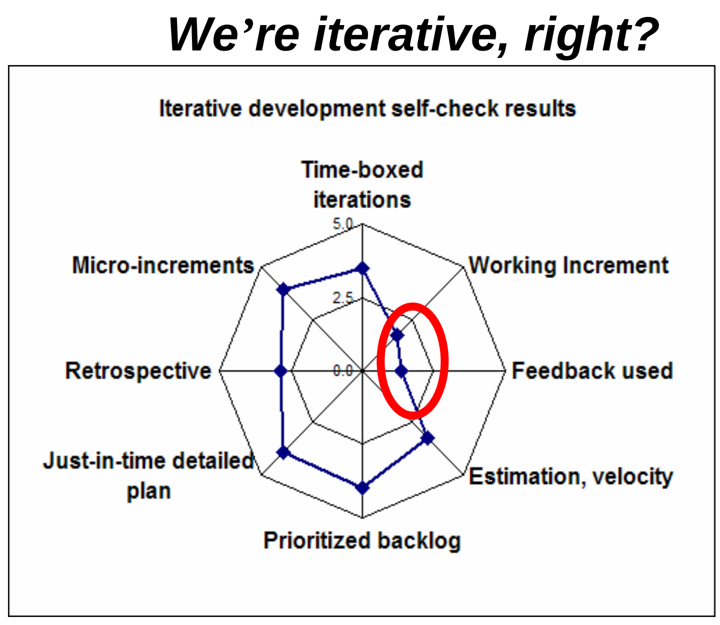


Figure 3. IBM® Rational® Self-Check for Software Teams enables teams within an hour-long session to assess themselves, graphically depict strengths and weaknesses, and take one or two improvement actions. The figure shows opportunities for improvement around automated unit testing and some aspects of iterative development.

To address the need for score-carding, we also need objective Control Metrics. These are provides through artifact-based metrics with associated guidance on how to interpret them to understand how well a practice is adopted. For example, "Build Health" (referring the health of a given "build" during the software project) is one of several metrics we can use for assessing the adoption level of iterative development: Teams new to iterative development typically only get stable builds in the last few days of the iterations, if that, while teams well versed in iterative development have stable builds throughout the iteration.

By analyzing our control metrics on a weekly or monthly basis, we can understand how well the adoption is going, so we can take the right corrective actions.

Minimizing change while maximizing the return

One of the innovations provided by MCIF is the ability to minimize change required for a given context. This is achieved by providing a library of well-defined discriminating software and system delivery practices that can be seen as atomic change units. These practices, such as *Iterative Development*, *Continuous Integration*, and *Use-Case-Driven Development*, can be adopted incrementally. Adopting the appropriate practices represent a meaningful change, leading to an observable and measurable impact in how people are working and in the outcome of the projects.

Determining which practices to adopt is not trivial; many factors impact the choices, including business and operational objectives, the SSD context, and the current maturity of the development organization. We have developed a variety of assets to help answer that question by analyzing each of those, plus other factors:

- Context Maps – A key asset for determining applicable practices, as described above.
- Value Traceability Tree Networks **–** A network capturing the relationships between the operational objectives and the change-set of chosen practices and metrics addressing those

objectives, as shown in Figure 4. This example focuses on the operational objective to improve quality. The left two branches deal with improving code quality, and the right two branches deal with improving business alignment. Two strategies are outlined for improving code quality: Increase Defect Prevention and Increase Defect Detection. The former suggests adoption of the practice Test-Driven Development, and the latter suggests adoption of the practice Iterative Development.

- IBM® Rational® Health Assessments for Software Delivery – This services asset allows us to analyze strengths and weaknesses in how organizations develop software, and produce an incremental and measurable roadmap for how to improve the business value of software delivery organizations, which focuses on the improvements with highest ROI first.

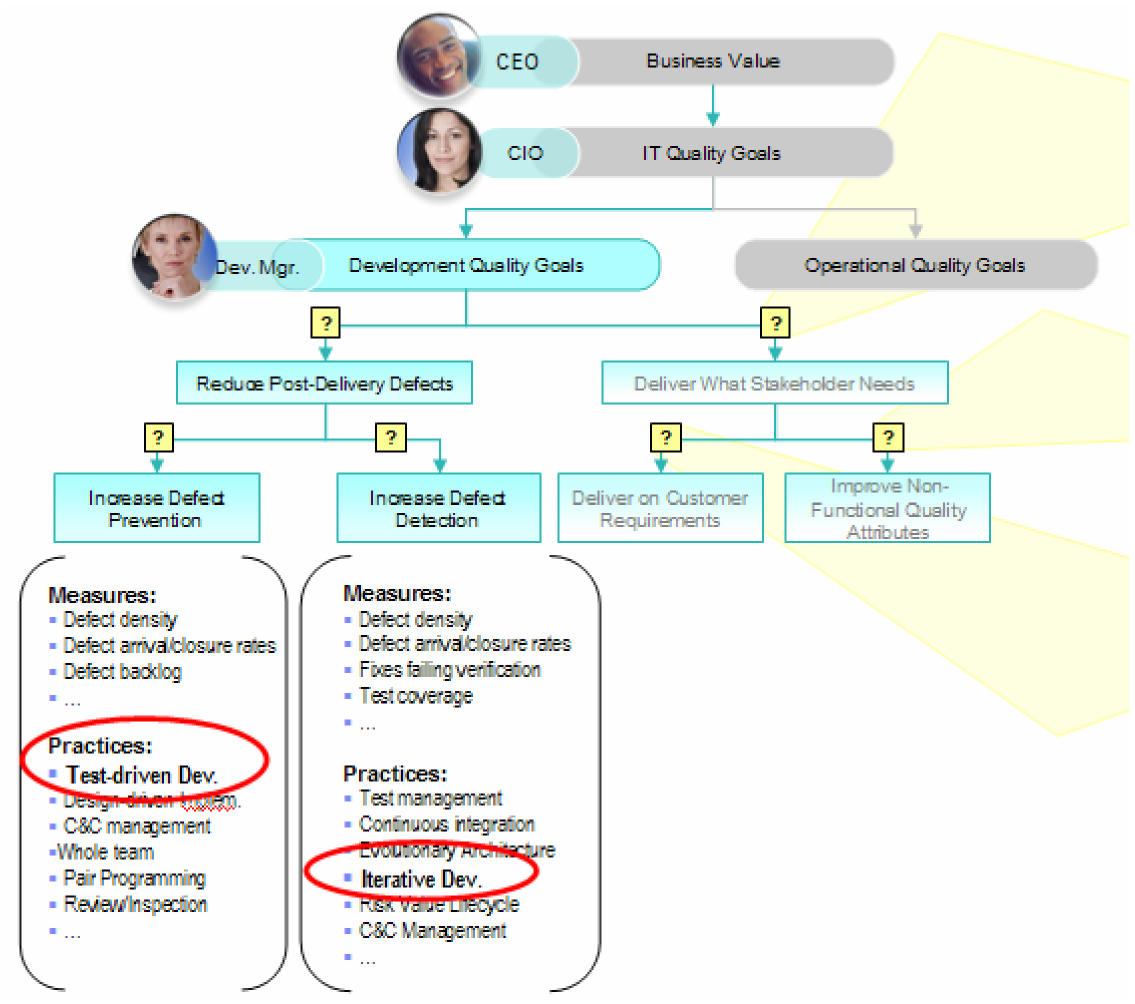

CEO
Business Value
CIO
IT Quality Goals
Dev. Mgr.
Development Quality Goals
Operational Quality Goals
?
Reduce Post-Delivery Defects
Deliver What Stakeholder Needs
Increase Defect Prevention
Increase Defect Detection
Deliver on Customer Requirements
Improve Non-Functional Quality Attributes
Measures:
Defect density
Defect arrival/closure rates
Defect backlog
…
Practices:
Test-driven Dev.
Design-driven Iterative Dev.
C&C management
Whole team
Pair Programming
Review/Inspection
…
Measures:
Defect density
Defect arrival/closure rates
Fixes failing verification
Test coverage
…
Practices:
Test management
Continuous integration
Evolutionary Architecture
Iterative Dev.
Risk Value Lifecycle
C&C Management
…

Figure 4: Linking operational objectives to practices and associated metrics.

Many organizations mistakenly try to make one process fit all circumstances. In our experience, the above type of analysis is required to enable you to drive the appropriate change to the right project types.

Applying the control framework with the IBM Rational organization

IBM Software Group is one of the world's largest software delivery organizations, with more than 25,000 developers producing software products driving in excess of $15B of revenue. Sample key business objectives for SWG include:

- Increase revenue
- Improve profit margin
- Increase market share

Sample key operational objectives for the software delivery organization to support the above include:

- Improving productivity / reduce cost
- Improving quality (business alignment, consumability, as well as code quality)
- Improve predictability
- Operate transparently

To address the above business and operational objectives, IBM Software Group decided to take on one of the world's largest agile transformations, incrementally moving our developers towards agile development practices. Key practices we continue to focus on include Iterative Development, Continuous Integration, Test-Driven Development / Automated Unit Testing, and Whole Team.[3] Further, the team has focused on deploying integrated lifecycle environments, enterprise reuse, business and operational level outcome metrics, as well as practice-level control metrics.

Productivity is also contextual: It depends on the sort of effort. For example a maintenance organization might measure cost per change request. An organization developing new features might measure cost per function points. Some find it sufficient to measure productivity in terms of the volume of code produced. At IBM, we take a return on asset (ROA) perspective to measure productivity as Revenue per employee and headcount per product release, as shown in Figure 5.

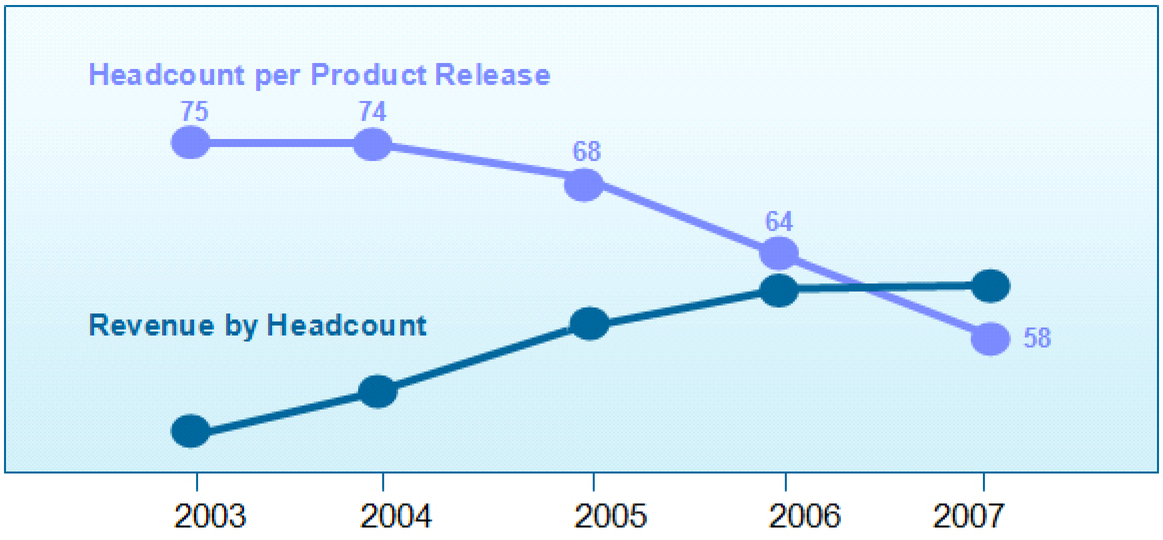


Figure 5: Since IBM Software Group started to focus on agile development practices in 2003, productivity measured as revenue by headcount and headcount per product release has improved substantially.

The IBM Rational organization has recently implemented an MCIF measurements solution using IBM® Rational® Insight. With the push to control costs, develop in an agile and scalable way, and handle data even across mergers and acquisitions, we knew that we needed the advantage of our software strengths to capture real-time metrics in order to better manage our business. Our goal was to develop a live, middle-tier, operational dashboard that would present data derived from multiple heterogeneous and inconsistent sources, including data from a variety of mergers and acquisitions. A key design point was to make the data easily consumable by executive level stakeholders by organizing it around key business and operational objectives, with drill-down for further analysis by product area or geography. Operationalized guidance defines the right actions, workflows, and policies to improve your measured results and be compliant with stated threshold values.

The following information describes sample on-screen analysis from that solution.[4] Figure 6 shows the top portion of the executive-level dashboard, which provides an overview of key products under development, such as IBM® Rational® Team Concert, IBM® Rational® Quality Manager, IBM® Rational® Requirements Composer, and IBM® Rational® Focal Point for Project Management. The dashboard in the upper half is a scatter chart visualizing data on four key aspects:

- Schedule (y-axis). Demonstrates variance to begin to illustrate some measure of predictability.
- Completion (x-axis). Projects are organized by percent complete, allowing executives to focus on projects nearing completion.
- Project health (bubble color). A score card is used to assess project health based on 12 factors shown in Figure 6.

Resource (bubble size). Provides executives with the ability to pay attention to the projects consuming the most resources.

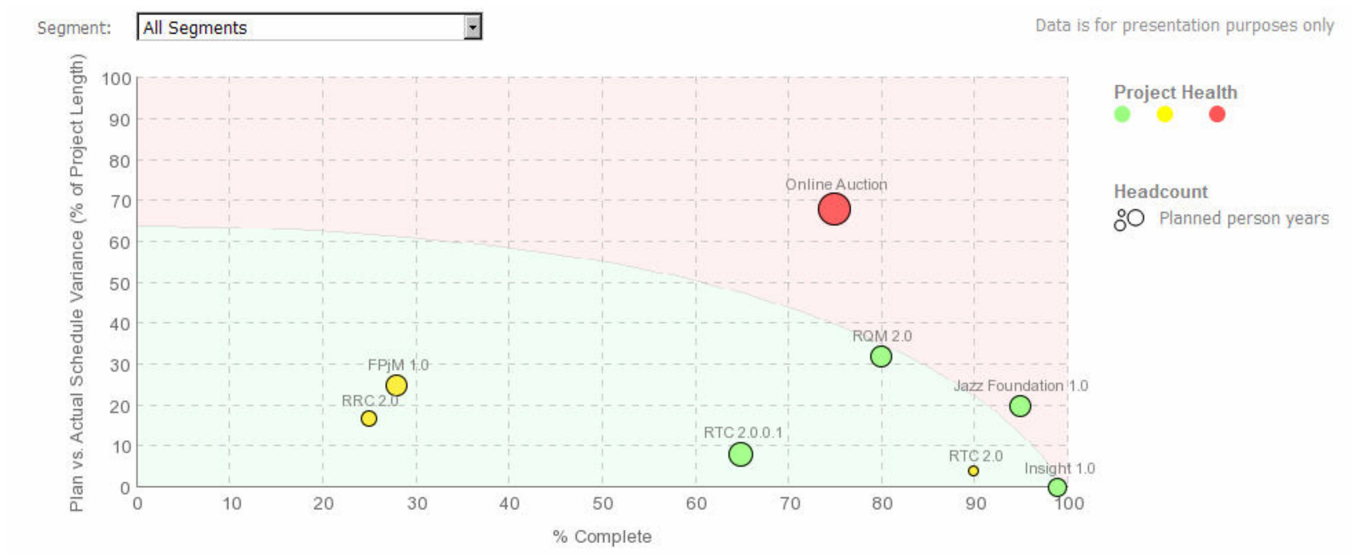
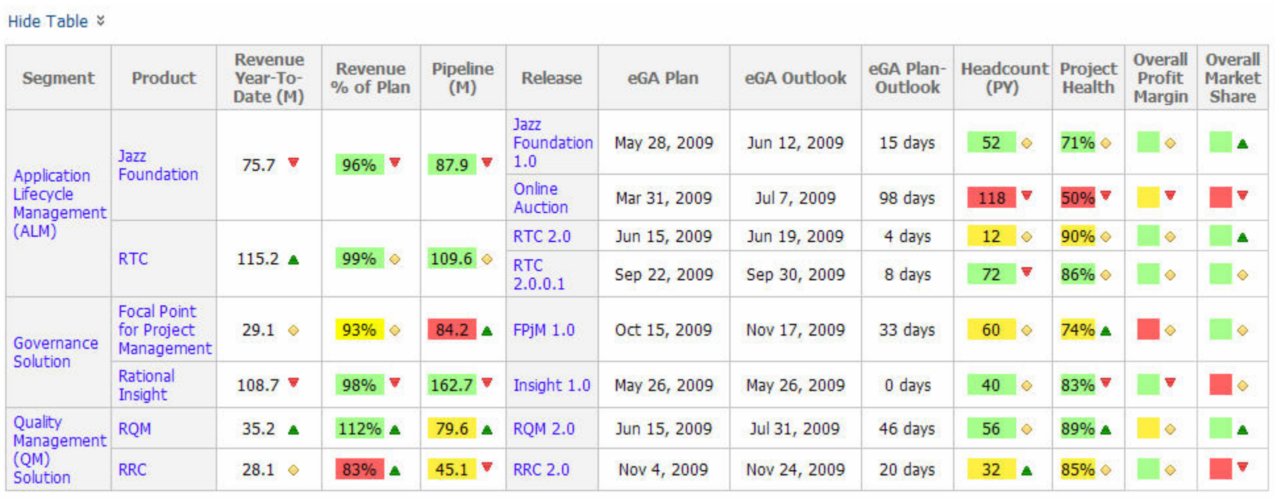


Figure 6. This executive-level dashboard draws information from many disparate data sources.

The power of the chart is really in the visual immediacy of the four factors together, which allows an executive both to gain an overall sense of all of the projects being tracked at one glance, and also to easily identify projects in the “red zone” based on a threshold curve overlaid on the chart. With almost no analysis time, an executive can identify and begin to drill down into detail on troubled projects, trusting that the exception path is built on accurate and live data.

Essential project info can easily be accessed through mouse over. Expanding the table allows you to see the details of the data. Clicking a bubble, an item in the table, or a country on the globe allows you to move through the top-level view into a more detailed set of data organized hierarchically from business objectives, which support operational objectives, to the detailed metrics underneath. For example, clicking Brazil in the globe (Figure 7) gives us the detailed view of that country’s headcount data sorted to only show the projects that are being developed in that country.



Hide Table ≚

Segment	Product	Revenue Year-To-Date (M)	Revenue % of Plan	Pipeline (M)	Release	eGA Plan	eGA Outlook	eGA Plan-Outlook	Headcount (PY)	Project Health	Overall Profit Margin	Overall Market Share
Application Lifecycle Management (ALM)	Jazz Foundation	75.7 ▼	96% ▼	87.9 ▼	Jazz Foundation 1.0	May 28, 2009	Jun 12, 2009	15 days	52 ◇	71% ◇	🟩 ◇	🟩 ▲
					Online Auction	Mar 31, 2009	Jul 7, 2009	98 days	118 ▼	50% ▼	🟨 ▼	🟥 ▼
	RTC	115.2 ▲	99% ◇	109.6 ◇	RTC 2.0	Jun 15, 2009	Jun 19, 2009	4 days	12 ◇	90% ◇	🟩 ◇	🟩 ▲
					RTC 2.0.0.1	Sep 22, 2009	Sep 30, 2009	8 days	72 ▼	86% ◇	🟩 ◇	🟩 ◇
Governance Solution	Focal Point for Project Management	29.1 ◇	93% ◇	84.2 ▲	FPjM 1.0	Oct 15, 2009	Nov 17, 2009	33 days	60 ◇	74% ▲	🟥 ◇	🟩 ◇
	Rational Insight	108.7 ▼	98% ▼	162.7 ▼	Insight 1.0	May 26, 2009	May 26, 2009	0 days	40 ◇	83% ▼	🟩 ▼	🟥 ◇
Quality Management (QM) Solution	RQM	35.2 ▲	112% ▲	79.6 ▲	RQM 2.0	Jun 15, 2009	Jul 31, 2009	46 days	56 ◇	89% ▲	🟨 ◇	🟩 ▲
	RRC	28.1 ◇	83% ▲	45.1 ▼	RRC 2.0	Nov 4, 2009	Nov 24, 2009	20 days	32 ▲	85% ◇	🟩 ◇	🟥 ▼

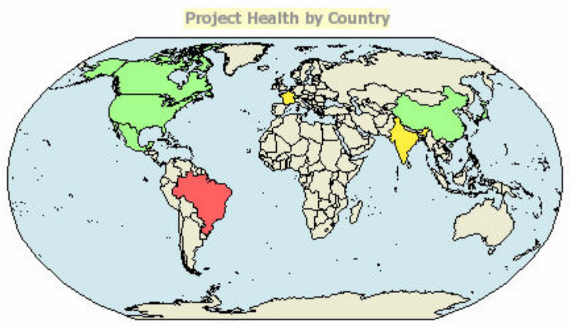
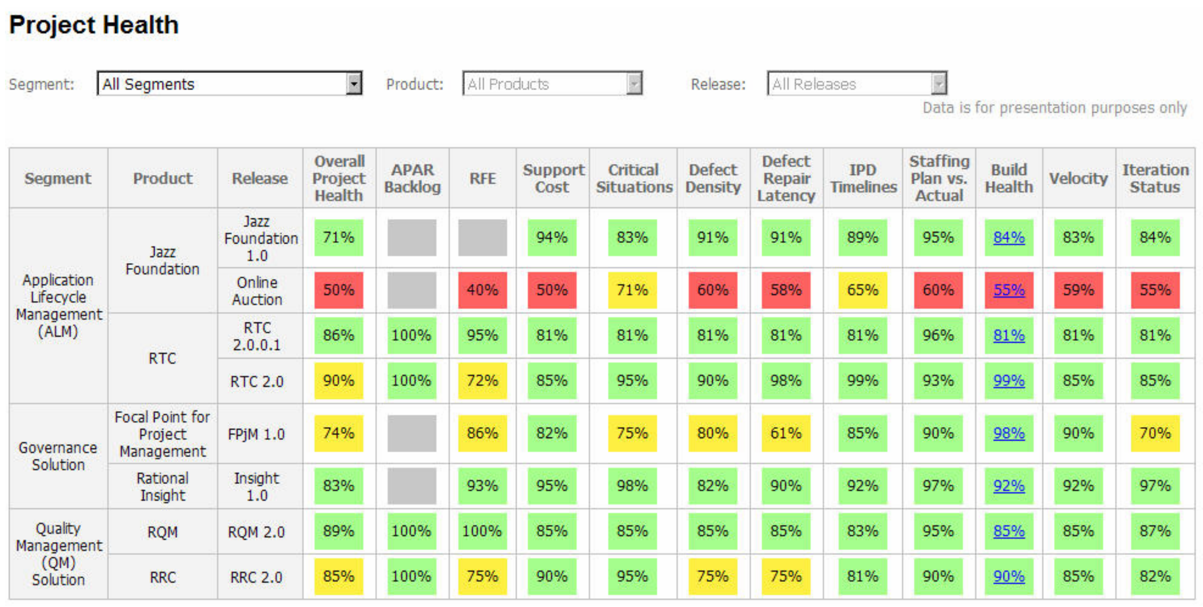


Figure 7. Expandable table of details and the globe view allows control of different levels of supporting data.

The drill-down on project health (Figure 8) is a composite made up of a variety of metrics that also functions as control metrics for key agile practices, including Iterative Development, Continuous Integration, and Test management. Each of these metrics allows further drill-down by clicking through the cells in the table to understand the individual causes of the troubled project.



Project Health

Segment: All Segments Product: All Products Release: All Releases

Data is for presentation purposes only

Segment	Product	Release	Overall Project Health	APAR Backlog	RFE	Support Cost	Critical Situations	Defect Density	Defect Repair Latency	IPD Timelines	Staffing Plan vs. Actual	Build Health	Velocity	Iteration Status
Application Lifecycle Management (ALM)	Jazz Foundation	Jazz Foundation 1.0	71%	▨	▨	94%	83%	91%	91%	89%	95%	84%	83%	84%
		Online Auction	50%	▨	40%	50%	71%	60%	58%	65%	60%	55%	59%	55%
	RTC	RTC 2.0.0.1	86%	100%	95%	81%	81%	81%	81%	81%	96%	81%	81%	81%
		RTC 2.0	90%	100%	72%	85%	95%	90%	98%	99%	93%	99%	85%	85%
Governance Solution	Focal Point for Project Management	FPjM 1.0	74%	▨	86%	82%	75%	80%	61%	85%	90%	98%	90%	70%
	Rational Insight	Insight 1.0	83%	▨	93%	95%	98%	82%	90%	92%	97%	92%	92%	97%
Quality Management (QM) Solution	RQM	RQM 2.0	89%	100%	100%	85%	85%	85%	85%	83%	95%	85%	85%	87%
	RRC	RRC 2.0	85%	100%	75%	90%	95%	75%	75%	81%	90%	90%	85%	82%

Figure 8: A drill-down view of project health showcases practice-level control metrics organized around business objectives.

Two other business objectives are also available in scorecard drill-down views with the supporting operational objectives and metrics.

Achieve profit margin: In order to meet this goal, you need to be able to answer a few operational questions:

- Are your projects being delivered on time in a predictable manner?
- Are you actively increasing revenue?
- Are you demonstrably reducing costs?

  This drill-down view gives you information around schedule and financials to help you understand how to achieve profit margin goals.

Improve market share: Increasingly, measuring the level of collaboration, reach, and transparency in your projects directly contributes to improving market share by increasing sales and potentially driving down support costs. As the Rational organization focuses on development transparency through efforts such as Jazz.net, we needed a way to capture this data. The Improve Market Share drill-down view captures information around our transparency level for each project, internal enablement, and greater collaboration aspects. Consider the current economy where travel restrictions are tight – how are you able to measure effectiveness in collaborating to determine if you are overcoming these hurdles?

As discussed, the ability to reduce time to market is critical as horizons are shortened in the current economy; the Executive Dashboard gives IBM Rational executives the ability to detect risks earlier and the MICF approach gives them the prescription to remedy challenged projects quickly.

Conclusion

Business process improvement in any domain, such as manufacturing or service delivery, takes discipline. Achieving software and systems delivery improvement has particular challenges due to the wide range of intrinsic uncertainty and diseconomies of scale. Nevertheless, the common 3-tiered framework can be applied by considering the context of the efforts and choosing and measuring practices in light of the context. With this framework you can set goals, measure against those, goals, and steer your organization to improvement that matters to your stakeholders.

Endnotes

1 See Walker Royce, "Successful software management style: Steering and balance," in *The Rational Edge*, March 2005, at http://www.ibm.com/developerworks/rational/library/mar05/royce

2 M. Cantor, "Investment Value of Development," forthcoming.

3 Collaboration across the extended team, including users, business stakeholders, development, services, and support. More on these practices can be found on the IBM Rational practices page at http://www.ibm.com/developerworks/rational/practices/index.html

4 We have altered data on these screen captures to avoid revealing sensitive business information.