Welcome to this IBM podcast, *Pushing the envelope: How to Optimize Quality and Performance in the software delivery lifecycle,* by Michael T Lundblad.

Michael Lundblad is a Program Manager within IBM software group responsible for driving strategic initiatives around IT lifecycle management, particularly software quality management.

During his 22 years in the IT field, Michael has been the IT Director for two US Marine Corps organizations, and consulted with healthcare, manufacturing, public and commercial organizations on IT application infrastructure, development, testing and operations issues.

Hello.

Today, I'm going to discuss with you the quality and performance aspects of governing software delivery and operations. We'll identify 4 recommended best practices for ensuring quality and performance, and some of the IBM advances that help make these possible.

But first, let's examine the situation today.

There is a lot of talk these days about IT governance and risk management. CIOs have for years sought to align IT priorities with business objectives. But what's different now?

According to a 2006 IBM survey, 65 percent of the world's top corporate CEOs plan to **radically transform their businesses through innovation** in order to compete and respond to what their markets need.

Enter **project and portfolio management systems** that help set up business cases to justify and manage IT projects. PPM solutions provide IT executives with business and technical insight or views into the progress of key projects. This helps them make **easier resource decisions** to influence the most important projects to their business.

The next piece of the problem is what I like to call actionable governance at the project level. Project managers need to know; in addition to how well their IT projects are going, **what are the business and technical risks to their success**. On a moment-to-moment basis they need to see cost, time, quality and resource metrics to determine if their projects are at risk.

So what's new about that? The answer is quality, and performance.

Any project manager you talk to will tell you that **project delivery is about cost, time and resources**. Managing scope has historically been solely about these three metrics, but what we're adding is the quality dimension. Can you see the roots of the issues we're addressing?

The future of business applications in software is **incredibly complex. Infrastructure issues** are huge, with new middleware advances & the internet, composite applications based on Service Oriented Architecture (SOA), and packaged application integration with

legacy systems. These all lead to quality and performance issues that spell disaster if the new system is released prematurely.

An analyst reported a couple of years ago that only 14 percent of new web-based applications meet end-user response time expectations on day one, while 35 percent said it worked fine in testing. What does that tell you about typical software performance discipline? **Most don't do load testing, and those that do, do it poorly.**

Additionally, the **cost to fix poorly performing applications** once they get into production can be in the millions of dollars once you estimate revenue losses. For example, in 1999 eBay rebuilt its entire infrastructure because of a major performance problem in production. That was years ago, but look how well they perform today.

So, what can we learn from mistakes made in the past?

Firstly, we have to understand that **traditional software development**, testing and management in siloed teams **will not work with the same time pressures** we just talked about. To remain competitive, businesses need to find ways to **improve quality while shortening time to market** for business-critical software. That's the challenge your software delivery team is facing.

So, software quality management must be continuous, governed and automated throughout the delivery lifecycle. Traditional testing simply validates that the software in development meets end-user expectations for functionality, availability and performance prior to deployment. You can't test in quality! Quality is a lifecycle concern for designers, developers, build-engineers and final testing. If testing remains the quality management function, IT will fail to meet business expectations.

What do we mean by continuous, governed and automated?

We used to think testing was the answer, so testing groups were formed often called quality assurance teams. Instead software quality has to undergo a cultural shift. It must be three things:

A) Continuous Integration is a new concept.

In 2006, the Software Engineering Institute (SEI) report, Performance Results of CMMI-Based Process Improvement stated that, of the organizations surveyed, those that implemented Capability Maturity Model Integration (CMMI) processes improved quality by 48 percent, while reducing costs by 34 percent and shortening schedules by 50 percent. Not that surprisingly, the process improvements SEI recommended include **early, iterative and continuous quality activities, referred to as "continuous integration".** This also requires a comprehensive, collaborative software fitness process model, upon which IBM has built its team-oriented Rational® software delivery platform.

B) Software delivery requires actionable governance.

Project managers must have a moment-by-moment grasp of quality metrics, so corrective actions can be initiated immediately. Furthermore, compliance issues and geographically distributed development and testing teams make this even more challenging. Project governance should **make software delivery run smoothly and quickly, and it should enable collaboration** among teams, regardless of their locations.

C) Anything that can be automated should be.

Individual practitioner activities, such as functional and performance testing, have been automated, and build engineers use timesaving build scripts. But more needs to be done to automate the process and steps between roles to improve organizational efficiencies,

save money and speed time to market. For example, checking code into a repository should trigger a build/smoke test, possibly including a functional regression and performance test. Even test lab management should be automated so machines are reserved, provisioned and queued up for running the test cases in an automated workflow process.

Can you picture how this new perspective would affect your own organization? To help you imagine how it might work for you, let's explore the four recommended best practices for ensuring high performing business-critical applications?

Firstly, focus on performance early in the lifecycle

Forrester has a great report called Performance-Driven Development that recommends focusing on performance from requirements to design, development, testing and production. **Every step has to consider the performance requirements from the beginning, and build to those requirements**. Compare that to how your organization currently approaches performance engineering?

Secondly, architect, design & build for performance

The application infrastructure should include all those things that middleware and database experts know about. Middleware architects look at thread pooling and load balancing, while software designers think about design patterns that perform well, and developers need the right tools to spot things like memory leaks, and to test components for performance before going any further. They can also instrument the applications to capture performance information when they are finally load tested.

Thirdly, develop a performance testing center-of-excellence

This is a key concept. How can you achieve it? Here are my thoughts:

- **Centralise all load testing efforts** for economies of scale on equipment, software, and skill usage.

- **Use a production-like testing environment** – this is hard, but one customer I know is using their disaster recovery site. An alternative is to use what I call a "wind tunnel" approach. This is a scaled down version of what's in production, capture the results and extrapolate what's needed in production using modeling tools.

- You'll also want to have a **separate, dedicated network** to isolate other traffic from the test.

- **Use tools for automated test lab provisioning**. 30 percent of all testing time is spent just setting up the lab for the test.

- **Use load generation and monitoring tools** that can view all application infrastructure components which could impact performance, and integrate the results into discernable problem diagnostics.

- I mentioned **modeling tools for capacity, scaling and configuration planning**, e.g. Hyperformix. These tools take load testing information like workload volumes with response time results, infrastructure usage (like the numbers of servers, sizing, network bandwidth), resource consumption (like memory and CPU utilization) to model the test, then extrapolate what a full production system should look like to achieve a given response time requirement.

- Also you'll need access to **expertise** like developers, designers, infrastructure tuners, business analysts, networks specialists, and so on.

Finally, the fourth best practice measure is to enable collaboration among development, testing & operations

The idea here is to provide **integrated tooling and processes** that link together these various groups so that during and after testing, problem diagnosis and repair is as fast as humanly possible. Things like importing log and trace data from the monitoring tools into the developer/tester workspace so they can isolate, diagnose and find source code issues in minutes instead of hours. The right tooling can stop finger pointing and **get these groups working together as a cohesive team**.

Feeling inspired? Let's look at a real life example of a customer following these best practices to give us an idea of what they can achieve.

One of our financial services customers in the UK jointly presented with me at the recent IBM Rational Software Developer Conference in Orlando Florida. They formed a center-of-excellence for performance testing that's part of their production support organization. They have a pre-production performance testing lab that mirrors their production environment and also is their disaster recovery site.

This team has a program where they consult with the business unit development teams on early performance testing. These development teams are responsible for running base-line transactional performance testing so that the **key problematic transactions are already tuned in isolation** before they get to pre-production performance load testing in a production-like environment with everything else running.

So this **saves them a lot of time**, and allows this latter stage of load testing to focus on finding things caused by peak operational loads, cross application conflicts, and other tuning concerns. This customer is also using the Tivoli Composite Application Management solutions for **isolating their bottlenecks** and providing **diagnostic trace information to developers** using Rational Application Developer and Rational Performance Tester which can resolve and repair any latent performance issues that come up. They also use the results for **estimating usage growth over time** so they know when they will have to upgrade or add servers for load balancing.

This has meant that the organization has (1) met their go-live dates, (2) spotted the impact of a new application under test upon other applications in the production-like environment, and (3) noticed the impact on resources (like servers or databases) shared by all other applications in the environment.

The IBM Rational Software Developer Conference gave me some other key insights that you might find interesting. For example, there were some exciting announcements that support the collaboration and automation themes.

First, IBM has always been very forward thinking in terms of leveraging open platform technologies to support collaboration that connects the various roles in software delivery and management via process and data integration. The future of our enterprise quality management platform will be based upon the "Jazz" platform, which will appear in something we call "Team Concert".

We will **plug all of our quality management offerings into this** to make the integration more seamless and provide an easier to navigate user interface for all the various contributors.

In addition we will have a **new test lab manager platform** based on something called "Styx".

Imagine a workflow engine that walks the testing group through identifying test assets in the lab, scheduling their use, provisioning the machines with the right software to run the tests, running the test cases, reporting the results for analysis, and releasing the test lab assets when complete. This means that **we can run more tests, and get new applications to market on time at less cost, and more** importantly, higher quality means less repair work after it goes live!

Second, IBM announced the acquisition of Watchfire, a testing solution for examining new software for potential security breaches; and Telelogic, which will **accelerate our client's ability to define, build, test, deliver and govern the delivery of complex systems.**

Finally, I noticed a marked increase in the numbers of attendees in our quality management track. Many of the sessions I attended filled the room. I see this as validation that our customers are vitally interested in IBM's leadership in the area of software quality. We also had record numbers of business partners and global system integrators in attendance. I heard Accenture had 76 people present.

So, we have come to the end of this podcast.

As we have seen, companies who truly want to transform their businesses have to apply a fresh look at how they approach quality and in particular best practices for performance engineering.

IBM is providing leading solutions for continuous integration, actionable governance, and process automation.

The four best practices we talked about today are those adopted by our customers who are pushing the envelope on their business applications to meet the expectations of their own customers in production on day one.

I hope you've found this interesting. If you'd like to find out more, you can visit

www.ibm.com/itsolutions/uk/developer/

where you can download whitepapers, view our webinars or contact us directly.

If you want to discuss any of the topics from this podcast with me in person, feel free to email me at Lundblad@us.ibm.com

Thanks for listening!