Rational software

Strategic quality assurance.

Steps to effective software quality assurance.

Dominic Tavassoli, IBM

Contents

Overview

The paradox of quality assurance is that, although "quality" is a key value for every organization, the actions taken to ensure it are often left until late in the lifecycle, when budgets are scarce, time is short and there is high pressure to deliver to the market. Software quality is particularly challenging as teams generally have little idea at the start of the project how much testing and debugging will be needed to produce a release that is "acceptable."

The challenge of quality assurance may be overcome by applying a bit of process and discipline, as organizations can collect data on previous projects and learn from experience what fault levels to expect. By capturing information such as fault origin, diagnosis, and cost, software quality assurance (QA) managers can pinpoint weak process areas and focus improvement budgets with measurable benefits. Quality assurance becomes part of a real corporate plan with objective focus and accountable results – which we call strategic QA.

Strategic QA helps organizations track faults across the software development lifecycle, providing QA managers with the insight necessary to predict fault levels and testing budgets and, more important, allowing them to focus process improvement budgets in an efficient way with provable benefits. Strategic QA not only helps organizations become more competitive, it also reaffirms the value of the QA manager in the project team and the organization.

Two best practices help implement strategic QA. First, enterprise-wide fault data collection is facilitated by rolling out an enterprise change management solution, which enables all teams to easily submit the right data through a Web interface, assists in analysis, and generates the metrics and reports. Organizations should also standardize on common process and lifecycle rules so that data is consistent and the improvement initiatives are effective.

Highlights

This paper details the challenges and solutions available for organizations wishing to improve their software development process, reduce costs, improve quality and increase reliability of planning. It also presents examples of companies that have successfully implemented strategic QA and, as a result, have become more competitive.

The quality assurance paradox

Customers expect software quality to be very high, yet we frequently hear in the news about bugs or design errors that cost time, money or worse – lives. As software development becomes more complex and innovative, and is created by distributed teams, delivering quality is increasingly difficult.

QA generally focuses on testing the software in the final stages of a project, when budgets are scarce and the pressure to deliver the product is high. To ensure quality, project managers must balance quality assurance with time to market.

How much testing is enough?

Very few project teams are able—or are willing—to predict how many bugs there will be in a release. Testing and fixing until all bugs have been found would be a very expensive task, if it were possible.

Software engineers have a motto: "There is always one more bug." The amount of testing needed to ensure quality is hard to quantify. Very few project teams are able – or are willing – to predict how many bugs there will be in a release. Testing and fixing until all bugs have been found would be a very expensive task, if it were possible. And is it reasonable? Most organizations prefer to deliver a product on time, with a few unimportant defects, rather than to deliver it six months late. Needless to say, each industry has a different level of acceptable quality; for instance, word processors and aircraft systems have different quality requirements.

Highlights

In order to provide correct budget and schedule estimates, QA managers need the means to predict fault levels.

What part of your process needs to be improved?
Many organizations often limit QA to testing the final software. Functional testing may be outsourced to low-cost countries, which makes ensuring quality "somebody else's problem." In fact, this activity is quality control – checking after the fact that quality levels have been met. On its own, quality control doesn't help teams build better software, a fact that can be very frustrating for the QA manager, as he is confined to the role of inspector of other people's work.

Enterprise change management solutions enable organizations to view which links of their process are the weakest and where to focus process improvement budgets.

Analyst reports indicate that the cost of fixing a defect increases dramatically the later it is found in the lifecycle—and the cost of correcting a fault detected during final system testing can be up to 200 times more than if it is found during the requirements phase. Very few organizations can confidently pinpoint the weakest links of their process, those where their budgets would have the most impact. Without objective process performance indicators, QA managers cannot focus quality improvement initiatives where they are necessary or where they will have the best return on investment, let alone prove their efficiency.

Predicting defects and errors in the software development lifecycle

Metrics for strategic QA
Defects and errors can be introduced and found during the different phases of the software development lifecycle (SDLC). The practices described in this paper are easily applied to all types of processes (e.g., waterfall, iterative) with

different phase names. In this document, we will consider the following phases:

- *Requirements analysis*
- *Design*
- *Implementation/coding*
- *Unit testing*
- *Integration testing*
- *System testing*
- *Customer usage (maintenance)*

The industry uses multiple terms and different standards to name software problems. In this paper, we will use the following definitions:

- *"Errors" are problems detected in the phase they were created in.*
- *"Defects" are problems detected after the phase they were created in.*
- *"Fault" is the generic term for a defect or an error.*

For instance, if during the design phase of a project, 10 faults were introduced, 7 of them were caught, and 3 had been in the design specification given to the developers to code from, then this phase would be said to have 7 defects and 3 errors.

Highlights

Two metrics are of particular interest in tracking and understanding quality in the SDLC:

- *Phase containment effectiveness (PCE) is related to the number of faults captured in a phase (represents how effective the process is at preventing problems from becoming defects).*
- *Phase screening effectiveness (PSE) is related to the number of prior 'escaped' defects captured in each (successive) phase.*

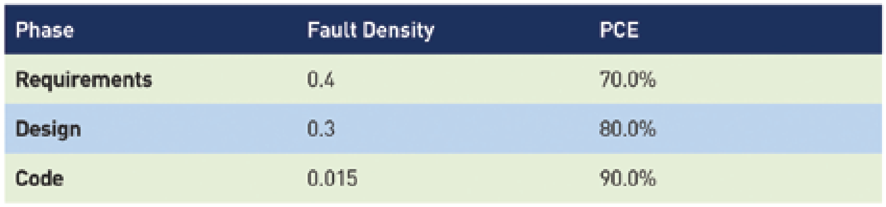
In our previous example, the PCE is 7/10. If later on in the process, out of the 20 faults present in a release delivered by the coding team and after unit testing, 14 were caught by integration testing and 6 were still present in the version delivered for system testing, the PSE for the integration testing phase would be 14/20, or 70 percent.

By capturing fault data on projects, organizations can estimate fault density (FD)—the number of faults present in a work product per size.

Predicting faults throughout the SDLC

By capturing fault data on projects, organizations can estimate fault density (FD) – the number of faults present in a work product per size, where size is defined accordingly for each phase. Requirements and design fault density may be expressed by faults per page, while coding fault density may be measured by faults per line of code (LOC) or by faults per function point.

Phase effectiveness metrics quantify an organization's ability to find and fix defects closer to their origin, before the cost of rework becomes too important. By implementing strategies to improve these metrics, organizations reduce the number of released defects and avoid project delays, improving customer satisfaction and reducing the cost of maintenance.

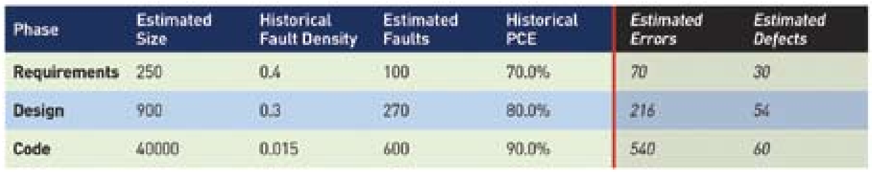


Phase	Fault Density	PCE
Requirements	0.4	70.0%
Design	0.3	80.0%
Code	0.015	90.0%

Table 1: Example values for quality prediction metrics

Historical project data should be collected to calculate these metrics for organizations. With FD, PCE and PSE values, the QA manager can statistically predict faults, errors and defects for each phase, based on the estimated size of the current release.

With the metrics discussed above, a QA manager can produce reliable fault estimates for a project. The table below shows an example of fault estimates that could result if the scope of the project were planned as 250 pages of requirements, 900 pages of design documents and 40,000 lines of code. The PSE metrics also help predict where these coding defects will be found.



Phase	Estimated Size	Historical Fault Density	Estimated Faults	Historical PCE	*Estimated Errors*	*Estimated Defects*
Requirements	250	0.4	100	70.0%	*70*	*30*
Design	900	0.3	270	80.0%	*216*	*54*
Code	40000	0.015	600	90.0%	*540*	*60*

Table 2: Example values for estimated faults

Highlights

With objective fault graphs, QA managers can analyze the performance of their quality processes within the company as well as benchmark this performance against other companies.

Historical fault data provides quality assurance teams with the information necessary to accurately predict fault levels across the SDLC. With objective fault graphs, QA managers can analyze the performance of their quality processes within the company as well as benchmark this performance against other companies.

Process improvement and the CMMI

Leading organizations have successfully attained return on investment (ROI) for process improvement initiatives by adopting the Capability Maturity Model Integration (CMMI) of the Software Engineering Institute (SEI).

One of the fundamental principles of the CMMI is the implementation across the organization of a standard set of processes that is continually improved based on objective measured feedback. In particular, the CMMI recommends the key practice of causal analysis and resolution. When implementing this practice, organizations identify the causes of defects and errors, and they take action to prevent them from occurring in the future. Causal analysis and resolution improves quality and productivity by proactively preventing the introduction of defects into a product.

Moving to strategic quality assurance

Capturing error and defect data across the organization

The data necessary for the QA metrics mentioned previously must be captured when faults are identified. This process is facilitated by the rollout of an enterprise change management (ECM) solution for consistent fault tracking. Web-based ECM solutions provide customizable forms to collect the relevant

Highlights

data for each fault. By providing a user-friendly interface with drop-down list boxes, organizations can ensure that quality feedback is easy for users to provide. Information to be collected for each error or defect should include:

- *Fault description.*
- *Category.*
- *The phase in which the fault was found (requirements, design, code, unit testing, integration testing, system testing, customer, not classified).*
- *How the fault was found (peer review, visual inspection, design model simulation).*

Extra information can be provided either by the submitter or later in the process, when the fault is verified and analyzed, and may include:

- *The phase in which the fault was introduced (requirements, design, code, not classified, prior release, third party).*
- *The phase in which the fault should have been detected.*
- *The cost of the fault.*

Enterprise change management enables organizations to implement a repeatable, documented and reliable process for capturing both fault data and change requests of all types.

Deploying enterprise change management across the enterprise

To successfully improve the way they produce software, organizations need to capture error and defect data in a consistent, centralized fashion. Enterprise change management, the cornerstone for sustainable compliance as well as causal analysis and resolution, enables organizations to implement a repeatable, documented and reliable process for capturing both fault data and change requests of all types, on software and hardware, from customers and the internal teams, urgent and minor. By offering a Web interface, ECM solutions support ease of adoption across the enterprise.

Highlights

Products that provide out-of-the-box, industry-proven processes are a natural choice for low-risk deployment.

Naturally, an organization wishing to deploy and enforce an enterprise change management solution must verify that it is scalable to its needs and flexible enough to implement a process that reflects the corporate culture and solves the identified challenges. The solution must also be capable of providing a common, consistent process across the organization. Products that provide out-of-the-box, industry-proven processes are a natural choice for low-risk deployment.

IBM solutions for strategic QA

IBM® Rational® Change software is a Web-based, workflow and change management solution for enterprise change management that helps simplify the change management process, enabling organizations to more consistently track defects and errors. With Rational Change, organizations can respond systematically to all types of change. This improves communication and collaboration throughout the development lifecycle and across the enterprise, allowing organizations to react to continuous change and potentially improving their productivity and time to market.

- *Enterprise change management helps enable organizations to respond quickly to change from both internal and external sources.*
- *Out-of-the-box forms, workflows, reports and metrics support best practices such as strategic QA, requirements-driven development, and process improvement initiatives such as CMMI and Agile.*
- *The Rational enterprise change process,, a ready-to-use process package is a standard part of the Rational Change product. It used by more than 900 users across more than 30 sites around the world, and is designed to unify software, electronics and hardware teams within a common workflow. This process has been certified for CMM Level 3 and 4 usage.*
- *Migration facilities from existing bug-tracking tools, both homemade and commercial, to IBM Rational Change.*
- *Lifecycle change management helps organizations coordinate all their development and management lifecycle tools, including IBM Rational DOORS®, IBM Rational Synergy and IBM Rational ClearCase®, for traceability throughout the entire development lifecycle.*

For more information

To learn more about IBM Rational software from IBM, contact your IBM representative or IBM Business Partner, or visit: **ibm.com/**software/rational

RAW14064-USEN-03