Using application quality metrics to improve the business value of applications

From The Rational Edge: Read how software quality metrics are becoming increasingly important to business health, and how software development organizations can begin incorporating quality metrics into their processes.

Why is the impact of application software quality growing?

The impact of application software quality has often been equated with availability and its dreaded converse, downtime. For applications such as reservation systems, CIOs have historically been able to compute lost revenue per outage minute. Four converging problems are elevating the importance of application software quality beyond its impact on availability: 1) the widespread damage resulting from application malfunctions, 2) the competitive need for business agility, 3) the dependence on suppliers, and 4) the exorbitant cost of application ownership.

Application malfunctions. Although many application failures only inconvenience a few customers, some threaten the business. Consider the impact when commercial Websites become inoperable, financial data corrupted, personal credit records breached, and bills miscalculated -- these are only a few of the application fiascos whose impacts range from lost customers to liquidated damages. Application quality metrics provide powerful diagnostics for detecting probable...
sources of application flaws before they cause havoc. Such metrics can provide input to enterprise risk analyses since poor quality applications represent substantial risk to business performance.

**Business agility.** The agility of a business is tied to the software quality characteristics of its critical applications. Today's accelerating need for businesses to adjust to new opportunities places a premium on rapidly modifying applications to support the changes in business processes and functionality. But new technologies do not necessarily fit on older platforms, and legacy applications do not necessarily port easily to new platforms. What's more, the demand for new functionality is often hampered by poor quality that has never been addressed. If the quality of an application erodes with age, the ability to rapidly implement new functionality declines as well -- just as the demand for the functionality accelerates. The structural integrity of application software must be sustained throughout its useful life.

**Supplier dependence.** Critical application software is increasingly supplied by external resources such as contractors, vendors, and outsourcers. The quality of externally supplied software is difficult to control, since service level agreements usually focus on post-delivery performance. Application quality metrics provide profound insight into supplied applications and provide much stronger governance over suppliers and their risks. Businesses also experience supplier risk when globally distributed internal groups develop or maintain different components of an application's software. The ultimate quality of the application service delivered to the business depends on ensuring that each group is working to the same set of quality objectives and that these objectives are evaluated when all components are integrated into a software system comprising the application.

**Application ownership costs.** Businesses are constantly pressuring their IT organizations to provide more service for less cost. Reducing labor costs through outsourcing has provided only a partial answer. Without constant attention to application software quality, today's state-of-the-art application can quickly devolve into tomorrow's legacy monstrosity. As an application ages, the percentage of time devoted to understanding its construction and fixing its defects increases relative to the percentage of time spent implementing new business functionality. Worse, the more difficult an application's software is to understand, the more likely developers will inject new defects with each modification. Unless the quality of an application's software is sustained, the relationship between its costs and the business function it delivers will continually erode.

**How do we define application software quality relative to the business?**

To satisfy the business demands created by these four converging trends, application software quality must be defined as more than the removal of defects. Defect removal defines application software quality in terms of test cases and is at best a lagging indicator of an application's fitness for use by the business. Rather, an application's quality must be defined in terms of the behaviors and attributes that provide the most effective service to the business. This is the approach taken by the Information Technology Infrastructure Library (ITIL), a collection of best practices with growing influence in IT departments globally. In fact, ITIL Version 3 defines **quality** as "the ability ...to provide the intended value". Similarly it defines **effectiveness** as "a measure of whether the objectives of [an application] have been achieved."
ITIL’s approach to quality requires that the business state its expectations for the service it receives from an application. Historically, business partners have struggled to state quantitative expectations for the business service provided by an application. In fact, they are often poor at anticipating many of the critical demands they will make of an application. It therefore falls on those who develop and maintain applications to anticipate business demands and proactively evaluate whether an application will be capable of satisfying current and future service expectations. Different measures of application software quality are required by different types of business demands such as agility, security, performance, accuracy, and availability.

Through extensive work with IT clients, CAST Software has clustered software metrics drawn from ISO 9126 and other sources into six characteristics of application software quality. These characteristics can be related to the business benefits delivered by an application as shown in Table 1. Taken together, these quality characteristics describe the overall health of an application -- the extent to which it is free from pathological conditions -- and its ability to service the current and future demands of the business.

Table 1: Application quality characteristics and their benefit to the business

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Description</th>
<th>Example business benefits</th>
</tr>
</thead>
</table>
| Transferability | Characteristics that allow new teams or members to quickly understand and work with an application | • Improves transfer of applications between teams for reasons of cost or effectiveness  
• Reduces learning curves  
• Reduces lock-in to suppliers |
| Changeability | Characteristics that make an application easier and quicker to modify | • Improves business agility in responding to market conditions or customer requests  
• Reduces cost of application ownership by reducing modification effort |
| Robustness | Characteristics that affect the stability of the application and the likelihood of introducing defects when modifying it | • Improves availability of the business function or service  
• Reduces risk of loss due to application malfunction  
• Reduces cost of application ownership by reducing rework |
| Performance | Characteristics that affect the performance of an application | • Reduces risk of losing customers from poor service or response  
• Improves productivity of application workers  
• Increases speed of making decisions and providing information  
• Improves ability to scale application to support business growth |
| Security | Characteristics that affect an application’s ability to prevent unauthorized intrusions | • Improves protection of competitive information-based assets  
• Reduces risk of loss in customer confidence or financial damages  
• Improves compliance with security-related standards and mandates |

Each of these application quality characteristics can be quantified in a series of metrics that can be presented in quality profiles and summarized in composite measures presented on management dashboards. They can be used proactively by development teams for early detection
and remediation of problems. They can be used surgically during maintenance to drill down to specific sections of an application that need correction. They can be used diagnostically to track changes in application software quality to ensure that the application's effectiveness for the business is improving. They can be used insightfully in deciding among tradeoffs in application costs and benefits.

**Why is application software quality harder to control today?**

Modern business-critical applications are no longer developed as monolithic systems written in a single, or at most two, languages. For example, a simple Java 2 Platform, Enterprise Edition (J2EE) application may be composed from multiple technologies, including JavaServer Pages/JavaServer Faces (JSP/JSF), JavaScript™, HTML for the presentation layer, XML for the coordination layer, Java for the business layer, and SQL for the database layer. Application software quality is therefore a multi-technology challenge with many quality problems occurring at the interface between technologies. The technical breadth of such polyglot applications exceeds the expertise of any single developer or project team because of the multiple languages, technologies, methods, and platforms involved.

Development decisions in complex applications involve tradeoffs between performance, maintainability, security, and other quality parameters that cannot be fully understood without comprehensive knowledge of interactions among application components and technologies. Consequently, the effectiveness of quality practices such as testing and peer reviews is constrained by the expertise of the development team. A thorough evaluation of application software quality cannot rely solely on human-dependent processes such as peer reviews and test case design. The evaluation of application system quality must be automated and performed at the systems level and must present objective, quantitative information about the quality-related attributes of the system.

Few modern business-critical applications are developed in a single project. Rather, the multiple subsystems that provide functionality, data management, user interface, Web access, and other capabilities are frequently developed in separate projects on separate continents by separate organizations. Most quality practices have been designed for use on and by a single project. Consequently, quality is usually evaluated at the level of an application subsystem. Application software quality must encompass all the subsystems and interactions required to provide the application's full business functionality if the quality of its service to the business is to be fully assessed.

**How should we use application quality metrics?**

Application quality metrics are versatile and can be used in a wide range of application lifecycle processes. Their uses can be grouped into four application lifecycle functions: 1) governance, 2) management, 3) evaluation, and 4) improvement.

**Governance.** This includes activities normally performed by executives or roles with authorities delegated by executives. These responsibilities derive from IT's accountability to the business and are frequently performed at the application portfolio level. Examples of governance functions that should incorporate application quality metrics include the following:
• Establishing portfolio-level application quality targets that best achieve application service objectives and compliance requirements
• Reporting progress in achieving application quality objectives and translating progress into quantified business benefits
• Making application investment and retirement decisions based on trends in their quality characteristics that predict costs and benefits
• Managing tradeoffs between business service objectives, application quality priorities, and business risks across an application portfolio
• Reviewing trends in application quality characteristics across an application portfolio to identify needed policies and actions
• Negotiating agreements on application software quality objectives to be achieved by contractors, suppliers, and outsourcers

Management. This includes the activities normally performed by project and application managers to ensure that application quality objectives are established, achieved, and sustained for each application. These activities oversee the technical work performed on each selected application and ensure that necessary remedial actions are performed. Examples of management functions that should incorporate application quality metrics include the following:

• Translating expectations for business service quality into quantified targets for application quality characteristics that can be prioritized as technical objectives
• Estimating and planning development resources, schedules, and skill levels based on application quality requirements
• Tracking the achievement of application quality objectives during development and maintenance to identify the need for corrective action that ensures early remediation of problems
• Providing feedback on application development and maintenance work to improve individual and team performance
• Predicting application life cycle costs, trends, and business impacts from trends in application quality characteristics

Evaluation. This includes activities conducted by the technical staff to measure or use application quality characteristics. These activities may be performed by those developing or maintaining an application, or by those performing shared technical functions such as configuration or release management. Examples of evaluation functions that should be based on application quality metrics include the following:

• Conducting regular evaluations of an application against quality objectives during development and maintenance activities to identify and prioritize remedial actions early to prevent expensive rework
• Drilling into metric data to guide technical decisions, especially those involving tradeoffs among quality characteristics, in developing or modifying an application
• Assessing application quality characteristics to ensure compliance with policies and standards
• Appraising the quality characteristics of application software submitted by contractors, suppliers, or outsourcers to ensure they satisfy contractual application quality commitments
• Assuring the quality of application components that can be reused in other applications or made available in component repositories

**Improvement.** This includes activities typically performed by the technical staff to improve their capability for meeting application quality objectives. These activities can be performed both at the application and portfolio levels. Examples of improvement functions that should be guided by application quality metrics include the following:

• Using feedback on application quality characteristics to improve personal or team knowledge, performance, and results
• Establishing quality baselines across the application portfolio that characterize its current state
• Investigating trends in application quality characteristics to identify needed improvements in methods, tools, processes, or training across the application portfolio
• Providing results from application quality assessments to contractors, suppliers, and outsourcers accompanied by targeted improvement objectives
• Comparing current application quality baselines against public benchmarks to identify gaps and improvement objectives

**How do we get started?**

The risks of big-bang approaches to organization-wide deployment are substantial. Our recommendation, based on extensive experience in improving application development and maintenance processes and deploying application technology, is simple: do a pilot implementation! By initiating the use of application quality metrics in a pilot with a single application or cluster of related applications, you get:

• Tighter control over the conditions of deployment and use
• A committed sponsor motivated to succeed
• Focused support sufficient to ensure success
• Credibility from a success story
• Lessons learned that help you adjust and improve deployment tactics

Even when an organization-wide deployment is planned, it is best to begin with a pilot implementation to learn how to deploy best practices and technology most effectively. To best exploit the advantage of a pilot the application quality improvement program should be started with a committed sponsor and project or application manager who are excited about the possibilities and willing to provide leadership for the deployment. Since the staff allocated to an application is usually busy with existing work, adding significant new responsibilities for collecting and interpreting application quality metrics can present undue risk for successful deployment. During the early stages of adopting application quality metrics, it may prove more effective to have external support for conducting the measurement and feedback activities.

As an organization gains experience on a few early deployments, it can begin integrating application quality activities into its standard processes and establishing the roles required to perform them. The collection of quality metrics can become a standard task integrated into build
procedures that are supported by a separate build or configuration management function. The interpretation and use of quality metrics may be aided by product and process assurance staff who are broadly knowledgeable in the technologies comprising most applications and the most effective use of the metric results.

The ultimate value of application quality metrics depends on their use by developers. The use of these metrics must be integrated into defined development and maintenance processes so that they can be planned and tracked. In an environment where the technical staff takes pride in the quality of the application service they provide to business partners, metrics can be perceived as a powerful aid to achieving professional objectives rather than a painful evaluation of one’s personal competence.

The more an organization uses measures to control its performance and results, the faster it increases its value to business partners and the greater influence it exerts. The complexity of business-critical applications has moved the control of quality beyond the capability of individuals, regardless of how talented they are, and requires the comprehensive insight provided by application quality metrics. The speed of their adoption ultimately translates into the agility and competitiveness of the business.