Uplift Quality with Requirements Driven Testing

Drive Quality According to Business Needs
Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Executive Summary</td>
<td>2</td>
</tr>
<tr>
<td>The Quality Challenge</td>
<td>2</td>
</tr>
<tr>
<td>Defects in Requirements Specifications</td>
<td>2</td>
</tr>
<tr>
<td>Incomplete Requirements</td>
<td>4</td>
</tr>
<tr>
<td>Elicitation Poorly Executed</td>
<td>4</td>
</tr>
<tr>
<td>Ambiguity</td>
<td>4</td>
</tr>
<tr>
<td>Personnel Involvement in the Lifecycle</td>
<td>4</td>
</tr>
<tr>
<td>Review of Requirements</td>
<td>4</td>
</tr>
<tr>
<td>Test Coverage</td>
<td>4</td>
</tr>
<tr>
<td>Tests Done at the End of the Development Cycle</td>
<td>4</td>
</tr>
<tr>
<td>Impact Analysis</td>
<td>5</td>
</tr>
<tr>
<td>Traceability</td>
<td>5</td>
</tr>
<tr>
<td>Application Complexity</td>
<td>5</td>
</tr>
<tr>
<td>Poor Scope and Change Management</td>
<td>5</td>
</tr>
<tr>
<td>Requirements Driven Testing—a Solution</td>
<td>5</td>
</tr>
<tr>
<td>Elicitation of Requirements</td>
<td>5</td>
</tr>
<tr>
<td>Earlier Test Involvement and Frequency</td>
<td>6</td>
</tr>
<tr>
<td>Test Coverage</td>
<td>6</td>
</tr>
<tr>
<td>Measurement</td>
<td>6</td>
</tr>
<tr>
<td>Requirements Driven Testing</td>
<td>7</td>
</tr>
<tr>
<td>Reducing Rework through Accurate Requirements</td>
<td>7</td>
</tr>
<tr>
<td>Definition</td>
<td>7</td>
</tr>
<tr>
<td>Accelerating Testing through Tight Requirements and Test Management Integration</td>
<td>7</td>
</tr>
<tr>
<td>Requirements-Based Management Visibility for Timely, Informed Decision-Making</td>
<td>7</td>
</tr>
<tr>
<td>RDT Summary</td>
<td>8</td>
</tr>
<tr>
<td>Overview of Applications Management Value Profile</td>
<td>9</td>
</tr>
</tbody>
</table>
Executive Summary

Most business and IT executives agree that any company that is able to rapidly deliver software of high and predictable quality with minimum budgets enjoys a significant advantage. However, practical experience shows that the challenges associated with software quality remain largely unsolved; despite massive investments in testing automation, only a small fraction of the IT application portfolio enjoys sufficient test coverage. It is also apparent that many applications eventually do not meet user requirements. These problems lead to poor user experience, lack of adoption, massive application rework—and ultimately loss of competitive advantage.

Chief among the reasons behind this problem is the fact that, in many software organizations, quality is handled as an afterthought. In such organizations, quality verification efforts typically begin only once they have completed the code. At this point, testing teams are under the gun to certify the application as quickly as possible. Many perceive such certification as a bottleneck that prevents deployment into production. In this environment, it is difficult to ensure that the requirements are correct; to properly plan tests; to ensure correctness, completeness and solid coverage of requirements; and to gain visibility into the various quality aspects of the tested application. It is no wonder that this proves to be a costly and frustrating exercise for stakeholders.

This white paper presents the fundamental principles of Requirements Driven Testing (RDT), a different testing process that:

- Is tightly coupled to application requirements.
- Is integrated through various activities throughout the software development lifecycle.
- Delivers measurably high, systematic and cost-effective requirements and test coverage.

By focusing on the above principles, RDT eliminates the quality bottleneck and ensures that quality and requirements are no longer treated as an afterthought.

The Quality Challenge

The state of quality within the software development lifecycle has shown some improvements, with a move from only 32% of projects being considered a success¹ to 37% as reported the following year². While an increase in success of projects by 5% may seem slight, it does highlight that organizations are focusing more on the challenges around the delivery of projects. Increasingly, organizations are trying to reduce cost and risk while increasing quality and agility.

This is partially addressed by the fact that the market has seen more organizations migrating to an Agile process over a traditional development methodology. Current analysts are showing that approximately 40% of projects now follow an Agile approach—which is being directly attributed to increases in project success.

Defects in Requirements Specifications

While the increase in the adoption of Agile methodologies will look to address this challenge, the uptake of Agile development is still slow. Agile development encourages the collaboration and review of business requirements at an earlier stage of the development lifecycle—providing defined review points to ensure that change can be considered and incorporated into development within shorter timeframes. The net result is early confidence from stakeholders that what is being developed will meet the needs of their organization.

The move to agility also increases an awareness to focus not only on functional testing, but also to consider non-functional testing, such as performance. Often, the requirement phase of testing will not focus on the non-functional aspects of requirements such as the system should respond within x amount of time, but more so on the functional aspect. This is an area that, if amended, will result in greater project success. This is highlighted by the fact that traditional projects and methodologies leave performance testing till later in the lifecycle—meaning that

---

if an architectural performance issue is identified, the cost and impact to the project can be huge. In addition, the knock-on effect could lead to delayed delivery of the release, a requirement re-architecture and massive reworks across the solution.

James Martin, in his work "An Information Systems Manifesto," states that 56% of all defects identified in software projects are introduced in the requirements phase of the software delivery lifecycle. Further, it states that about 50% of requirements defects are the result of poorly written, ambiguous and incorrect requirements. The other 50% of requirements defects can be attributed to requirements that were simply omitted.

Five issues contribute to the poor quality dilemma in the requirements arena. These issues are:

- Incomplete requirements.
- Elicitation poorly executed.
- Ambiguity.
- Testers, developers, business analysts, architects, etc., are involved too late in software development lifecycles.
- Review of requirements.

Industry analysis reports that the cost of defect correction increases according the stage of the development lifecycle where the defect is found. Projections show that defects found in production may be 100% more costly than if identified during the requirements phase. This should motivate businesses to improve build quality during the early requirement phase of a project.

This reflects the fact that defects are found in the following areas:

- Requirements: 15%
- Design: 20%
- Development: 40%
- Testing: 15%
- Production: 10%

If not detected and addressed early enough in the design and development cycle, the costs can be catastrophic and felt in so many ways:

- **Revenue loss.** When software under-performs or fails to meet the needs of its user community.
- **Increase in support costs.** When applications are not reliable and user complaints are directed to a support organization.
- **Lost productivity costs.** When software under-performs or fails completely.
- **Brand damage.** When the user community complains publicly, especially on social network sites, or, worse, changes brands.
- **Liability costs.** When software fails to perform as promised.
Incomplete Requirements
Authoring requirements can be a daunting task even for seasoned professionals. The task of uncovering all of the business, functional, non-functional and technical requirements is a large effort. Requirements are often specified poorly by domain experts who may lack the skills to define consistent, accurate, and testable requirements. Tooling, if used, may not allow for efficient validation and/or review of the requirements with the business, technical, and functional users. Incomplete and/or inconsistently applied procedures for requirement gathering further escalate this issue.

Elicitation Poorly Executed
How are requirements extracted from business users? Most often, there is a session(s) where business users explain what they want the product, change request, or new service to accomplish. Requirements are often vague and lack the precision for analysts, developers or testers to actually use them. This is not an issue of an inadequate business user—rather it is a symptom of the lack of requirements elicitation methods designed to pull requirements in an organized, repeatable, and reproducible process. The lack of this discipline leads to incomplete, vague, missing and incorrect requirement statements and user stories.

Ambiguity
How often have you seen pages and pages of requirements in a voluminous document? It’s difficult to understand requirements embedded in paragraphs. There are often multiple requirements in a single paragraph, and the requirements are vague. Descriptive words like quickly, normally, likely, should, etc. lead to requirements without clarity and conciseness. The ambiguity of the requirements results in rework and clarification cycles.

Personnel Involvement in the Lifecycle
Business users, analysts, modelers, architects, developers, testers—the team for the development of a solution—are often not integrated into the cross-functional team that starts at the beginning of a project and stays together for the entire lifecycle. The notion of involving everyone may not be possible in some organizations due to priorities. Minimally, testers and developers must be involved earlier in the cycle to mitigate rework cycles.

Review of Requirements
The review process associated with requirements, that is, peer review, business stakeholder review, approver review, is seldom done. Often, the process is somewhat ad-hoc and is inconsistently applied across and within projects. The sheer volume of requirements can be overwhelming and the time that it takes to adequately review the requirement assets is ignored.

Test Coverage
With the increased adoption of Agile methodologies, comes an increase in collaboration between teams, specifically development and test. Reducing the siloed activities increases the visibility of test activities and brings quality into the development cycle at a much earlier stage.

With teams now working side by side, organizations will see a marked increase in collaboration of activities and reuse of assets.

No longer does the responsibility of quality sit solely with the test team—but increasingly the responsibility of testing is falling on the development team as well. Part of the reason is that development has the ability to consolidate the test assets within the test management solution and provide open visibility as to the quality of builds.

In so doing, quality activities become part of a continuous integration approach to testing, ensuring early visibility of issues and a resultant decrease in defect resolution costs. Focus on testing should also not remain focussed on functional aspects of an application—but on the non-functional aspects, such as performance testing.

The following issues make good test coverage exceptionally difficult to accomplish:
- Tests done at the end of the development lifecycle.
- Impact analysis.
- Traceability.
- Application complexity.
- Poor scope or change management.

Tests Done at the End of the Development Cycle
The move to Agile also means that more teams are no longer leaving test to the end of the cycle, but rather including it as part of the iterative approach to development. This shift has enabled quality to be built into applications at an earlier stage to reduce the cost of defects. Encouraging organizations to incorporate test earlier in the development lifecycle, enables quicker response to resolution of issues and increased delivery quality. While test traditionally remains at the later stages of a project, functional issues may be resolved, but non-functional issues such as performance remain hard to resolve.
Impact Analysis
Much of “what” needs to be tested is dependent on village knowledge—relying on the subject matter expert with years of experience in the product line to simply know what to test and what results are acceptable. The lack of formal, programmatic, repeatable impact analysis (manual or automated) contributes to the introduction of post-production defects, incomplete testing and identification of downstream impacts to simply “changing one line of code.”

Traceability
Similar to impact analysis, and of equal if not greater importance, is the concept of bi-directional traceability. The importance of linking requirements to other requirements, i.e. business requirements to functional requirements to user requirements, cannot be understated. Likewise, traceability extends from requirements to models, source code and test cases. A clear, organized tracing structure is often missing in organizations and is one of the most important concepts that should be utilized to reduce rework and increase test coverage.

Application Complexity
The complexity of modern applications (and in particular distributed applications) makes it very hard to cover all of the possible scenarios. The sheer number of alternative paths through an application and the inherent non-functional requirements associated with each possible outcome, results in large numbers of interdependent test cases that extend the execution timeline to an unacceptable level. These factors result in further delays in project timelines.

Poor Scope and Change Management
Scope creep is a fact of application development. Most projects (even those with good controls) experience scope changes as a result of emails, hallway conversions, etc. While these suggestions are made with good intent, they often become part of scope, which in turn, impacts development, requirements quality, rework, and most importantly, testing. Inadvertently, the requirements change without authorization or visibility back to the business or forward to the testing team. This is a prime cause of application defects. Along with poor scope management, undisciplined change management adds to the confusion.

Tooling that supports the flagging of changes becomes increasingly important, with updates to requirements necessitating organizations to quickly react and understand the impact to quality process and effect to timeline and scope.

Requirements Driven Testing—a Solution
Requirements Driven Testing (RDT) has four major methods contained in its framework. These methods are elicitation and visualization of requirements, earlier test involvement or frequency, test coverage, and measurement. It sounds easy enough, but it’s important to note at this stage that ideally your priority should be to focus on high-risk testing.

Your ability to identify business risk and prioritize against the requirements and test will enable the focus of all activities to be managed according to criticality and business risk. Extending the requirements approach to include risk delivers an advanced capability to manage and control what you test, when you test, where you tests and how you test.

Of course, we realize that it’s virtually impossible to perform 100% of all testing, so it’s important to determine the priority and drive of what you do. It is through a disciplined approach to these methods that RDT can be effectively used and deployed within an organization. Let’s examine each of the methods in more detail.

Elicitation of Requirements
You’ve heard the old adage that “a picture is worth a thousand words.” This saying correlates strongly to the idea of requirements elicitation and visualization. The ability to extract requirements from the business user in a non-paragraph based approach facilitates clearer requirements, a high degree of collaboration and an interactive environment for the analyst, developer and tester to work with the business user.

WHERE WOULD YOU INVEST ENERGY?

Figure 3. Where would you invest energy?
Requirement gathering is not the same as requirement elicitation. Good requirements are not just collected from the business user, as would be indicated by the name requirements gathering. Requirements elicitation is non-trivial; you can never be sure you get all requirements from the user and customer by just asking them what the system should do.

Visualization is an extension of requirements elicitation. It is the component that enables business users to “see” their requirements in three dimensions rather than as a sentence or paragraph. Visualization is especially useful for application or products that contain presentation layers; however, the concept applies to all projects. The powerful benefits of RDT begin to emerge when organizations combine visualization with elicitation techniques, ensuring developers, testers and analysts are involved earlier in the software development lifecycle.

**Earlier Test Involvement and Frequency**
RDT promotes a testing process that is integrated throughout the entire software development lifecycle. As soon as requirements, design and code artifacts are ready, they are reviewed and tested against best practices, requirements, use cases and test cases.

Testing becomes a parallel activity to the development process, a constant pursuit that spans all roles and makes all stakeholders aware of quality objectives. As a result, testing is no longer a bottleneck activity, performed after code is delivered under extreme time pressure. Many defects are detected at earlier development phases when it is much cheaper to fix them. Additionally, there are significantly fewer surprises when code is delivered.

**Quality is not a discrete or isolated function; it is an ongoing and cumulative effort.**

**Test Coverage**
RDT facilitates the method of expanding test coverage coupled with the use of impact analysis and traceability. As requirements are constructed, and testers and developers are involved in the up-front requirement cycles, an increase of test coverage can be expected. This is primarily due to the fact that RDT breaks down the old paradigm of “throwing the code over the wall” in the testing world.

Impact analysis and test coverage are self-evident. The impact may be widespread, encompassing not only unit test, but also system, integration and regression testing efforts. RDT seeks to expand the proficiency and, ultimately, the quality of software projects.

RDT supports methodical and systematic design of test cases to ensure predictable test coverage. Following a rigorous process, RDT attempts to facilitate test case design that does not rely on the skill or experience of specific testers. Rather, it strives to inject repeatability and method to the testing process in order to make test coverage more predictable. It also attempts to apply various optimization techniques to produce the minimum number of test cases required for sufficient test coverage. By doing this, RDT makes the testing cycle more rapid and manageable.

RDT stands on two legs:
1. quality assured requirements and test cases
2. using both in all phases of the software development lifecycle.

**Measurement**
RDT promotes a quality process that can be managed and improved through measurement. Throughout this process, multiple measures help quantify the status of deliverables and activities. This helps managers and process experts oversee quality initiatives across the IT application portfolio.

Metrics can be applied at different levels of scope and at different points in time. In general, three levels of metrics exist—portfolio metrics, in-flight project metrics, and post-mortem project metrics (Forrester Consulting).

RDT supports all levels of metrics gathering, but the “sweet spot” for RDT is in the in-flight project metrics area. The metrics are gathered while the project is still underway. In-flight metrics are most useful to the development and testing teams, although others may also be interested. In-flight metrics can enable decisions that actually change the course of a project and affect post-mortem project metrics.
Requirements Driven Testing
Micro Focus has a better way to approach software delivery—Continuous Quality Assurance (CQA). Instead of trying to test defects out at the end of the project cycles Continuous Quality Assurance leverages requirements management and automated software quality solutions to ensure software quality and eliminate defects from the beginning to the end of the lifecycle. One of the core concepts of Continuous Quality Assurance is that requirements are at the heart of the application delivery process.

Reducing Rework through Accurate Requirements Definition
Software teams have always struggled with requirements. Reaching an early consensus on how software should look and behave reduces the risk of unnecessary rework; developers have an accurate impression of what needs to be built and quality is jumpstarted early.

The Micro Focus® requirements definition solution enables enterprises to create visual, interactive simulations of the software in minutes. Using a simple to use drag and drop interface, non-technical users can easily create working prototypes with clickable menus and functions, just like in a fully working piece of software. Web-based access to requirements also enables teams to collaborate, review and offer rapid feedback on concepts and changes before they go into production.

Micro Focus’s visual requirements definition allows team members to understand precisely what needs to be done from the start, vastly reducing the risk of waste and rework further down the line.

Accelerating Testing through Tight Requirements and Test Management Integration
The Micro Focus requirements management solution offers open bi-directional traceability for integrating requirements definition and management with other application delivery applications. Because our requirements definitions have embedded user scenarios, they can be automatically transformed into test cases. This automatic transformation allows the acceleration of testing during projects and project iterations.

Further to that, it is important to note that tight integration ensures that time and effort is better spent on working on the business critical areas of an application—through visibility of business risks, priorities and status of requirements. Ongoing change of requirements can impact the test assets, so automatic flagging of changing requirements helps provide fast impact analysis capabilities.

Requirements-Based Management Visibility for Timely, Informed Decision-Making
Beyond the productivity benefits in creating test cases from requirements, the linkage that is created between them helps align testing to business value. Using Micro Focus solutions to automatically link requirements to tests enables QA Managers to answer questions on how well testing has covered new software and existing functionality. Open Test Management solution Micro Focus® Silk Central™ allows you to align your business requirements regardless of source, e.g., Word, Rally, Doors, RequisitePro or Micro Focus Caliber with the tests, enabling central collaboration and traceability of, from requirements to test to execution and issues.

The Micro Focus solutions leverage business attributes like criticality and other important characteristics, enabling QA Staff to effectively prioritize testing and assign appropriate coverage. It also allows managers to answer questions on the value of assigning more or less resources to projects or test cycles. Application of Quality goals, which Silk Central allows you to define in your organizations terminology and applies your risk factor or calculation, ensure you drive testing to meet quality goals. Central reporting and analytics provide the real-time visibility and traceability across the lifecycle.

While almost all organizations have increased their drive to automate testing, large percentages continue to perform manual testing. If you’re one of those organizations, the Micro Focus test management solution, Silk Central, allows you to use the quality goals and risk factors that support the requirement and risk-based approach to testing. In so doing, Silk Central provides an advanced manual execution planning capability, allowing you to drive your manual test planning from the quality goals and criteria defined against your requirements, through to which resource will perform the tests, which timeframe the tests will be performed in and what environment the tests will be run against. This is all coupled with status
reporting and analytics to provide you with clear direction and visibility to monitor your actual progress against your planned progress.

Because of its ability to link quality effort to business value and manage business risks, the requirements to test linkage is a core requirement of software delivery today and is a key capability of the solution.

With Micro Focus’s range of market leading requirements management and automated software quality tools, customers are better able to ensure quality throughout the lifecycle, from the definition of the first requirement through the development of the system and up until the point of release.

The following Micro Focus products enable RDT:

- **Micro Focus Silk Central** is an open test management solution that reflects the way today’s organizations work. As a scalable, flexible test management engine, it ensures that development teams deliver higher quality software into the hands of their users faster, ensuring greater value in a shorter timeframe.

- **Micro Focus Silk Test™** delivers the efficiency and productivity benefits of functional test automation while mitigating the classic challenges of implementing automated GUI testing such as time, budget and resource constraints. Successful automation comprises manageable development effort, reusable and easily maintained test assets, the ability to run the automation whenever and wherever it’s needed, plus quick and accurate results analysis.

- **Micro Focus Caliber** allows organizations to manage changing requirements throughout the development lifecycle while maintaining traceability to related assets such as test cases, source code and design or specification documentation. By providing effective coverage and impact analysis for proposed changes, Caliber ensures that organizations have increased control over and focus in their development efforts.

Silk Central helps prioritize, execute and demonstrate how test efforts respond to initiatives such as risk-based testing. It has a flexible, straightforward and easily understood mechanism for defining risk mitigation requirements and the quality goals determined by the business, using its own terminology. It shows the time cost of meeting those goals and helps test planners select a test set efficiently. Reports then show how testing has responded to the risk mitigation and other quality goals, ultimately enabling the go or no go decision on the release readiness of a project.

The solution also provides an interactive requirement definition platform that helps organizations through visualization of requirements to understand the software projects they commission prior to investing in them. Simulation provides a foundation for innovation and acts as a force multiplier for downstream IT development investments by ensuring the right “thing” gets built every time while also eliminating avoidable rework.

**RDT Summary**

The central focus of this white paper is RDT—why it is important, what potential areas of the software development lifecycle are addressed by RDT, the benefits of RDT and the implementation strategy that may be adopted to introduce RDT within the organization. By integrating quality activities with the overall software development lifecycle, RDT can help
organizations to systemically deliver predictable levels of software quality. With RDT, quality is not treated as an afterthought, nor are testing efforts perceived as a bottleneck in the software development lifecycle.

RDT is a discipline and as such may “change the game” for many organizations. Implemented correctly, RDT can be a high-powered catalyst for an organization to reduce waste and rework. Disciplined practices, like RDT, can reduce defect introduction rates. Change adoption by an organization depends on factors such as “the organization’s existing software maturity level and the staff’s and organization’s willingness to operate within a highly structured software culture.” (Software Defect Reduction Top 10 List, Boehm, Basili, Jan. 2001). RDT is not an invasive concept, rather a realignment of existing resources throughout the software development lifecycle to be utilized earlier and more frequently to drive out defects and increase quality of an organization’s products and services.

How can this improvement be measured? To know how much you can improve, an organization needs to have baseline figures from which to base improvement. Gathering that information can be a daunting task for many IT organizations, and that is where the Applications Management Value Profile (AVP) may be able to assist.

**Overview of Applications Management Value Profile**

Micro Focus has been gathering statistics from customers over the last fifteen months using a model called the Applications Management Value Profile. The model assesses organizations on three levels—financial, critical business issues and adherence to industry best practices in the disciplines of project management, quality management, requirements definition and management, and software change and configuration management. The AVP enables organizations to objectively assess their current processes in these areas and provides a recommended roadmap for future software investments based upon projected business value.

The results thus far show the top four critical business issues facing customers:

- Customer satisfaction
- Application delays and cost overrun
- Feature and functionality concerns
- Reliability

It is not surprising that these critical business issues have direct and indirect ties to requirements and testing. Absent from the success equation is the notion of Requirements Driven Testing. The concept is quite simple—approach testing from a standpoint of testing early and often with testers involved earlier in the development lifecycle, test with hard facts and systematically substantiated cases and paths, and test with measurement and improvement in mind.

For more information on the AVP or to assess your organization’s maturity in requirements or testing, email: requestavalueprofile@microfocus.com