Martin Kochloefl

(E-Mail: martin.kochloefl@microfocus.com)
ist Senior Consultant AMQ Division bei Micro Focus (IP) Limited. Nach seinem Informatikstudium arbeitete er zunächst als freiberuflicher Programmierer und Trainer, war dann als Projektleiter und Senior Consultant und Implementation-Manager bei Merant, SQS und Borland tätig. Martin Kochloefl bringt 25 Jahre IT-Erfahrung mit, davon 10 Jahre als Berater und Projektleiter. Seine aktuellen Schwerpunkte liegen bei der Definition und Umsetzung von SCCM-Prozessen und -Lösungen sowie im Bereich IT-Management & Governance.

Bringing Requirements to Life to Drive Collaboration and Agreement
The Critical Role of Automated Functional Testing in Enterprise Environments

„This year's results show a decrease in project success rates, with 32% of all projects succeding, 44% were challenged and 24% failed".
The Standish Group CHAOS Report 2009

Executive Summary
Software development projects suffer most when changes in requirements set off a cascade of delays, revisions, and rework. This happens all too often, as existing processes for establishing requirements are ad-hoc and inefficient, making it difficult for business people to communicate their needs to technology teams. This leads to miscommunications and insufficiently defined requirements that drive up development costs and delay projects.

To address the root cause of the problem, you need a more effective requirements definition (RD) process that reduces rework, speeds development, and leads to dramatic time and cost savings that maximize the return on your development investments.

Caliber TeamDefine addresses these challenges by enabling you to bring business and IT together early in the process to quickly iterate on a working model of the desired software. Because you can quickly bring requirements to life in real time and showcase proposed user interfaces (UIs) and software functionality before you start developing actual code, you can get to the right requirements faster – and with greater confidence – minimizing rework later. Now business and IT professionals can work visually and collaboratively to discuss, negotiate, agree, and publish requirements in ways that everyone understands: using visual models that replicate how the final product should look, feel, and behave.

Communication Breakdowns In The Requirements Definition Process

Successful enterprises are agile and responsive to changing business conditions and customer needs. They have applied best practices to everything from data center management and security to CRM in an effort to maximize efficiency and resource utilization. In addition, they have invested heavily in application development because custom software, which is created by programmers who work closely with business analysts, is often the best way to match functionality exactly with company needs.

But when it comes to best practices and efficiency, development projects are often less successful than other IT initiatives. According to research from the Standish Group, many projects are plagued by a combination of delays, reduced feature sets, budget overruns, and cancellations.

The fact is, planning and launching a development project can be complicated – even if you have lots of experience. All the stakeholders – business analysts, coders, quality engineers, legal and managerial staff, end users, and support staff – must collaborate on a growing set of requirements. And this is no easy task, as it's notoriously difficult to motivate everyone to participate, secure time in everyone's calendars (especially executives) for numerous meetings and requirements reviews, and gain agreement on what's needed.

Even when you are able to drive collaboration across key stakeholders, it hardly guarantees project success. Why? Because too often, stakeholders walk away from meetings with different interpretations of written requirements and what to expect in the finished software. In a survey by Cutter Consortium, analyst Alexandre Rodrigues found that, "projects are time compressed,

intensive, and mission-critical efforts with poorly defined requirements." [Rod01] Respondents cited several problems that torpedo the timely delivery of software projects, including:

- Unstable, constantly changing requirements (66%)
- Poor requirements specification (55%)
- Client behavior, such as approval delays, requirements changes,
- and poor communication (42%)

As these statistics suggest, traditional requirements definition (RD) processes aren't effective in helping stakeholders – both business and technical – elicit accurate, complete and well-validated requirements up front so that 'everyone's on the same page'. Because there's no single, shared vision of the new software that everyone agrees upon, development teams are set up to fail.

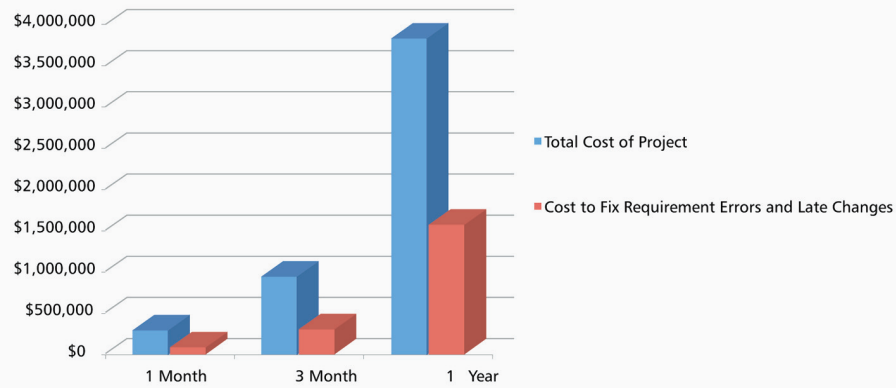

Figure 1: Cost of neglecting requirements on a 25-person project by project duration

Wrong Requirements? Lose money.

These RD failures – once viewed as just part of the software development process – are not acceptable in today's economic environment because they drive up costs unnecessarily. Sources such as James Martin's "An Information Systems Manifesto" indicate that 56% of software defects can be traced back to requirements as their source. At the same time, the cost to fix defects due to failed requirements accounts for 82% of the effort to fix defects. Multiple sources state that roughly 40% of any software project is dealing with rework. This means that for a typical, three-month project staffed with 25 people and requiring 16,000 hours of effort, about 6,600 hours of the project is rework. Of those 6,600 hours, roughly 5,400 hours (or 82%) would involve fixes due to failed requirements.

When you consider the cost of these hours in terms of salaries or contractor payments, the impact of failed requirements is stunningly high. If you assume the average, fully-loaded cost of a software engineer is paid $80 per hour, your company could be spending over $1.7 million for a year's worth of projects because of rework due to failed requirements.

But higher development costs are only the tip of the iceberg. The follow-on effects of project delays caused by poorly defined or incomplete requirements can be considerable. For example, the longer it takes to bring a software product to market, the harder it is for your business to respond to changing technologies and customer needs. You also increase the risk of missing market windows and losing months of revenue. Given that you have limited development resources, project delays on one project can also delay the start of other planned projects, which pushes out the return on investment for the next initiative as well – either as revenue or business savings. These impacts, while harder to measure, are critically important to consider.

At the same time, you run the risk of developing products of poor quality or with functionality that's incomplete or not what users need. Research indicates that for a typical application, well over half of software features and functions are rarely or never used. Imagine how many more valuable products you could develop by channeling those resources to new initiatives – simply by not building functionality that won't be used anyway?

Understanding The Root Cause

But why are changes in requirements so disruptive, problematic, and costly anyway? Consider a typical software development project. For example, imagine that a company is building a new call center application that will have many different screens and give users fast access to data from a variety of databases. After working through a traditional, document-driven requirements definition process with a group of power users of the old system, the business analyst leading the team captured and accurately documented all requirements.

But the company ended up with a system that only power users could use. The screens were filled with abbreviations and acronyms known only to power users, and they were extremely data intensive rather than graphical. The UI was simply unintelligible for common users. In addition, users found that the new screens no longer matched their existing processes; they had to constantly jump around in the application, which drove down agent productivity. To build a useable system, the company would have to rebuild the entire front end of the application, which would involve gathering a team of common users to provide input into requirements.

This is a classic "IKIWISI" (I Know It When I See It) situation; stakeholders participate in a requirements definition (RD) process that captures their needs and wants in words – but they don't know what they want until they see it – Or really have a chance to try it. At this point, coding that's already been completed must be modified and reworked to reflect the changed requirements, which drives up development costs, decreases customer satisfaction, pushes out ROI on the application itself, and prevents developers from working on other priority projects.

Investments In Optimizing Requirements Development Yield Big Returns

Given the high cost of accommodating changing requirements after coding has begun, investments in ways to address RD weaknesses can yield big returns. As shown in Figure 2, fixing problems at the RD stage has a significantly greater effect on

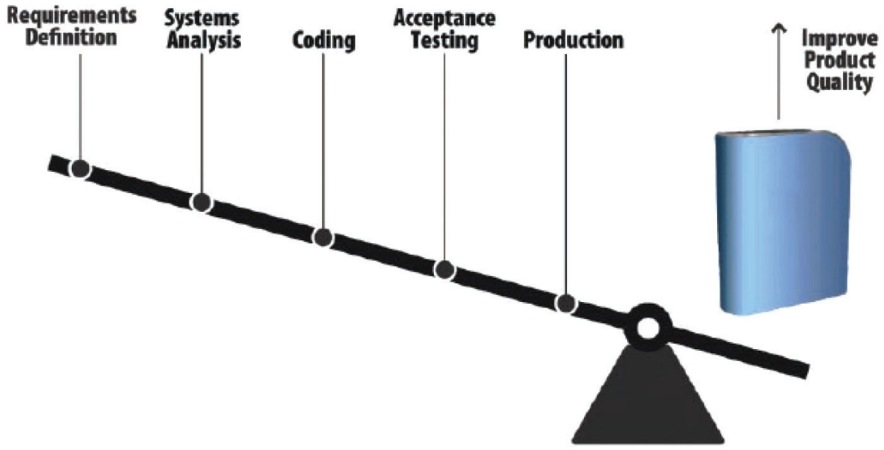

Figure 2: Leveraging the Power to Improve

improving the final software product than changes made later in the process. Equally important, they can be fixed with less time, effort, and cost.

So where does it make the most sense to apply effort? Certainly not during production or acceptance testing. Where would you choose to apply leverage? As shown in Figure 2, in the continuum of software development challenges, you get the most "bang for your buck" by getting requirements right the first time. To return to our call center application scenario, imagine the outcome for the business if the various types of end users had been able to sit in front of a simulation (also referred to as a prototype or model) and test out the proposed UI and process support before coding even began. Common users could have raised red flags early regarding usability and process hindrances. All parties could have defined requirements clearly and collaboratively at the outset. And the developers would have built the right product the first time.

Simulations And Visualizations: Keys To Getting Requirements Right The First Time

What's needed to make this scenario possible? You need a way to bring your requirements to life so everyone involved in RD can jointly visualize what the software needs to look like, feel like, and do – down to a good level of detail. Accurate, adaptable software simulations make this possible. When you have everyone engaged in a rapid, collaborative prototyping process, you're able to capture accurate, complete, and mutually-agreed to requirements up front. Your workgroups can:

- Engage in free-flow discussions around working models to build true consensus
- Ensure requirements are understood and formally approved by the right parties
- Greatly reduce the amount of wasted time, effort, and money on rework downstream in projects

Companies are discovering the benefits of software simulations in low-cost ways – for example, through paper prototyping. With paper prototyping, you enable representative users to perform realistic tasks by interacting with a paper version of the user interface. Users "pretend" to use the paper screens to experience, understand, and provide feedback on how the intended interface will work – before any coding begins.

Using Simulations To Elicit And Validate Requirements

Paper prototyping works fine for small projects and teams that are co-located. But it's not always sufficient for larger, complex systems and fragmented teams that need to collaborate virtually. Visualization software – which supports rapid, on-thefly prototyping – enables you to create working models of software during RD meetings with stakeholders located anywhere. Because business stakeholders can actually see proposed functionality, UIs, and workflows in front of them, they can react to what they see, discuss needs collaboratively, suggest changes, and then see them come alive – right before their eyes. In other words, everyone involved in defining the requirements can collaborate and experiment interactively until they see a working model that meets everyone's needs and expectations. The end result is that business stakeholders finish the RD process with an accurate, shared understanding of requirements – and they have a visual (rather than just text-based) way to communicate those needs to the IT professionals building software.

Once there's a stakeholder-approved simulation of the software, then:

- The requirements can be communicated unambiguously to the development team, with the most-needed features placed high on the list of developer priorities.
- Developers can actually "test drive" software before development begins and ask clarifying questions as needed. All of this happens before a dollar is spent on actual development.
- Test cases and release criteria can be established based on these requirements.
- Developers can begin coding with confidence that that their final product will satisfy everyone's needs and expectations.

Too Much, Too Little – Or Just Right: The importance of "Pragmatic Fidelity" when evaluating software to support visualizations and rapid prototyping, make sure you choose a product that helps you achieve "pragmatic fidelity" – in other words, simulations that are suitable to your audience and prototyping needs. Overachieving and underachieving on visualizations and prototypes can negatively impact your RD process:

- Overachieving prototypes, while impressing people right off the bat, often require a significant investment in time and encourage people to make design decisions far too early in the process.
- Underachieving prototypes leave dangerous latitude in interpretation of key features and introduce ambiguity into your RD process.
- To achieve the "just right" level of detail, select a prototyping medium that offers users the

opportunity to create simulations of pragmatic fidelity. Your tool should allow you to include business logic, integrate real-life data and basic functionality, and help control the urge to take simulations too far too early. Perfect is often the enemy of "good enough."

Enabling Software

Software supporting these types of "interactive" requirements definition processes must:

- Be easy for non-technical team members to learn and use
- Enable users to create working simulations – or software prototypes – for stakeholders to review
- Support the capture and management of feedback
- Be consumable by a variety of stakeholders representing different disciplines
- Make it easy for distributed teams to share simulations and collaborate virtually

Realizing A Chain Reaction Of Benefits

When requirements are defined visually and accurately and ratified by all stakeholders before coding begins, efficiencies accrue throughout the project's lifecycle. Specifically:

- The design and coding tasks can easily reference the models already agreed upon by the business analysts, users, and IT staff, so misunderstanding and miscommunications are greatly reduced.
- Because there will be fewer clarifications and adjustments later on, rework is reduced.
- Features are developed with respect to priority, so unnecessary or superfluous elements are less likely to bog down the project's progress.
- Testing and QA can be planned against the ideal prototype – rather than against what's been built. As a result, quality teams stay focused on the right test cases, testing as intended.
- Test cases can be created more quickly and earlier in the lifecycle – and even concurrent with the development process. This saves time and money compared to the traditional practice of creating realistic test cases after there's a working application to base them on.
- End user satisfaction increases because the finished product matches their initial expectations, requests, and needs.
- Because there's less reworking, development teams can complete projects faster and be available to start the company's next project sooner.
- Time to revenue accelerates (and time to market) because your software is marketed and sold earlier than before.

Industry Showcases: Proving The Value Of Rapid Prototyping

Requirements definition and management (RDM) products aren't just increasing efficiency in theory. Major development organizations have implemented similar technologies that let them virtually model, prototype, and simulate their proposed products and innovations, and they have realized impressive results. The software makes it faster, easier, and cheaper to change critical assumptions, test out a new idea, and iterate freely on versions until the product meets all key stakeholder requirements.

Michael Schrage, author of the book, "Serious Play: How the World's Best Companies Simulate to Innovate," refers to these softwareenabled activities as hyperinnovation. He notes that:

"Cisco Systems Inc., for example, uses software to simulate network architectures for customers deciding what kind of digital nervous systems they want to build for themselves. The Boeing Company and the Chrysler Group rely on a CAD/CAE package called Catia to prototype their airplanes and automobiles; Goldman Sachs Group Inc. depends on Monte Carlo simulations to stress-test its derivative and synthetic security innovations. American Airlines Inc. and the Federal Express Corporation digitally redesign their operations research models to manage their just-in-time logistics and pricing models."

As these examples illustrate, simulations and rapid prototyping have been used to great effect – and it's finally being leveraged to optimize software design and development.

Using Application Simulations To Support Agile Development Approaches

It's well known that requirements management doesn't often appeal to agile developers – especially purists. And it's true that engineeringcentric approaches to RDM are linear, slow, and can appear to hinder innovation. But working without a clear understanding of what requesters want, agile developers often end up going back to the drawing board more than once, frustrating stakeholders and driving up costs along the way.

Working software simulations are an ideal solution to bridge the gap between agile development teams (the builders of software) and requesters of software who have very specific requirements. For example, they enable:

- **Agile teams to innovate and get to accurate requirements quickly:** Software simulations provide developers with a better understanding of requirements so they can get started quickly and execute with confidence. Equally important, they can be used as part of a prototype-driven innovation process (via repeated, quick prototyping) that supports the agile development process [Sch06].
- **Product owners to get feedback on their ideas quickly – even when working with highly distributed stakeholders:** Too often, business stakeholders aren't co-located with development teams – especially as more companies use offshore development to cut costs. But this hinders agile development processes because it just exacerbates the communication issues that initially drove up development costs and encouraged outsourcing in the first place. Software visualizations can help by enabling developers to create a proposed, working simulation and make it accessible using a URL so that business people (and even their peers) can provide feedback in real time.
- **Mature agile development teams to start coding without written requirements:** Mature agile teams can potentially use simulations to see and understand requirements well enough to quickly start coding with confidence.

Caliber TeamDefine: The Best Tool For The Job

When your company is ready to implement a comprehensive, simulation-driven RD strategy, Caliber TeamDefine is the right choice. Caliber TeamDefine enables you to build a simulation, or prototype, based on

requirements gathered collaboratively with stakeholders at all levels. It's so easy to use that you can build prototypes and make on-the-fly changes as you sit in front of stakeholders and respond to their feedback. Now business analysts, end users, and developers can each review and verify the information in the "language" best suited to their work style: through visual models, or simulations, that everyone understands.

Once a project's requirements are simulated and validated through Caliber TeamDefine, they must be tracked and managed throughout the development process. CaliberRM integrates seamlessly with Caliber TeamDefine. Through CaliberRM, requirements can be used and traced by Silk testing tools, as well as a myriad of other downstream technologies.

Integration with CaliberRM provides other benefits as well. While CaliberRM can manage any type of requirement and link all requirements to downstream development artifacts, it can't detect whether or not you've entered poor requirements. Now you can use Caliber TeamDefine to ensure that the requirements you're managing in CaliberRM are accurate and reflect what end users really need. In other words, you ensure that CaliberRM helps you build and test the right software the first time.

Leveraging The Benefits Of An Open ALM Solution

With its Open Application Lifecycle management (ALM) initiative, Micro Focus creates a framework where tools work together flexibly to support any workflow style or process already in place and integrates seamlessly with a company's existing management solutions – both commercial and open source.

Open ALM enables customers to streamline and improve existing processes by integrating with lifecycle tools and adding a management layer that aggregates information from all the components and presents a unified platform for reporting and asset trakking. In fact, Micro Focus offers solutions that support the four critical ALM processes involved in successful software delivery: project and portfolio management, requirements definition and management, change management, and lifecycle quality management. These tools work together and with third-party software to automate and improve software development, testing, and deployment for customer organizations.

Summary

Caliber TeamDefine has intentionally been designed to be a powerful and affordable solution that meets the needs of distributed project teams working on simple-tocomplex software projects. With an accurate software simulation developed with Caliber TeamDefine, you can bring your requirements to life – and turn the RD process into a dynamic, collaborative process that ensures you get requirements right the first time. As a result, you can:

- Save time and money throughout the design, development, testing, and QA stages of development
- Ensure your developers focus on features and functions that end users want and will use
- Meet production deadlines and budgets by minimizing unnecessary rework
- Free up resources to begin work on other strategic initiatives for the business

Micro Focus's complete portfolio of flexible and powerful project management, testing, and deployment tools ensure that application development projects break free from traditional bottlenecks and delays. Because processes and projects become streamlined and efficient, you can ultimately drive growth and greater success for your business.

Source

[Rod01] Rodrigues, Alexandre, "Project Goals, Business Performance, and Risk." Cutter Consortium e-Project Management Advisory Service Executive Update 2(7), 2001.

[Sch06] Bringing Design to Software. Schrage. May 8, 2006 version.

About Micro Focus

Micro Focus, a member of the FTSE 250, provides innovative software that allows companies to dramatically improve the business value of their enterprise applications. Micro Focus Enterprise Application Modernization and Management software enables customers' business applications to respond rapidly to market changes and embrace modern architectures with reduced cost and risk.

For additional information please visit: www.microfocus.com