serena™

Requirements Management

The Proven Way to Accelerate Development

Requirements Management

The Proven Way to Accelerate Development

Contents:

serena™

Requirements Management

The Proven Way to Accelerate Development

By Karl E. Wiegers, Principal Consultant at Process Impact and Sandra McKinsey, Senior Product Marketing Manager at Serena Software, Inc.

Abstract

Requirements management is an often underutilized discipline in software development. Yet the economic benefits – lowered development costs, faster development and better products that delight customers – have plainly been demonstrated in study after study. In this paper, renowned engineering process expert Karl Wiegers makes the case for requirements management as an integral part of any product lifecycle. Metrics are provided to show the positive effect of requirements management, from individual developers to the business to the customer. Serena® RTM® is examined as an example of how to bring effective requirements management into product development.

Many software problems arise from shortcomings in the ways that people acquire, document, agree on, and modify the product's requirements. Typical problem areas include informal information gathering, implied functionality, inadequately defined requirements, and a casual change process.

Most people wouldn't ask a construction contractor to build a custom $300,000 house without extensively discussing their needs and desires and refining the details progressively. Homebuyers also understand that making changes carries a price tag. However, people blithely gloss over the corresponding issues when it comes to software development. The widely quoted CHAOS reports from The Standish Group relate the consequences of casual approaches to requirements engineering. Year after year, lack of user input, incomplete requirements, and changing requirements are the major reasons why so many information technology projects fail to deliver all of their planned functionality on schedule and within budget.

Nor do requirements issues only plague the early stages of a software project. One study at a large defense contractor found that 54 percent of all the errors were discovered *after* unit testing was complete, and that 45 percent of these were requirements or design errors (Davis 1993). Other studies also indicate that 40 to 60 percent of all defects found in a software project can be traced back to errors made during the requirements stage. Nonetheless, many organizations still practice ineffective methods for this essential project activity. The typical outcome is an expectation gap, a difference between what developers think they are supposed to build and what customers really need.

Requirements Stakeholders

Nowhere more than in the requirements process do the interests of all the stakeholders in a software or system project intersect. These stakeholders include end users, customers acquiring the product, requirements analysts, project managers, developers, testers, regulators and auditors, manufacturing staff, sales and marketing, and field support or help desk staff.

Handled well, this intersection can lead to exciting products, delighted customers, and fulfilled developers. Handled poorly, it is the source of misunderstanding, frustration, and friction that undermine the product's quality and business value.

Requirements are the foundation for all the subsequent software development and project management activities. Therefore, all stakeholders must be committed to following an effective requirements process. This process includes activities for:

- Requirements development (eliciting, analyzing, specifying, and validating requirements)
- Requirements management (dealing with the requirements once you have them in hand)

Developing and managing requirements is hard! There are no simple shortcuts or magic solutions. But a toolkit of techniques, coupled with tools to store and manage requirements, provides a powerful defense against the expectation gap.

When Bad Requirements Happen to Nice People

The major consequence of requirements problems is rework—doing something again that you thought was already done. Rework can consume 30 to 50 percent of your total development cost (Boehm and Papaccio 1988) and requirements errors account for 70 to 85 percent of the rework cost (Leffingwell 1997). As illustrated in Figure 1, it costs far more to correct a defect that is found late in the project than to fix it shortly after it was created (Grady 1999). Preventing requirements errors and catching them early, therefore, has a huge leveraging effect on reducing rework. Imagine how different your life would be if you could cut your rework effort in half!

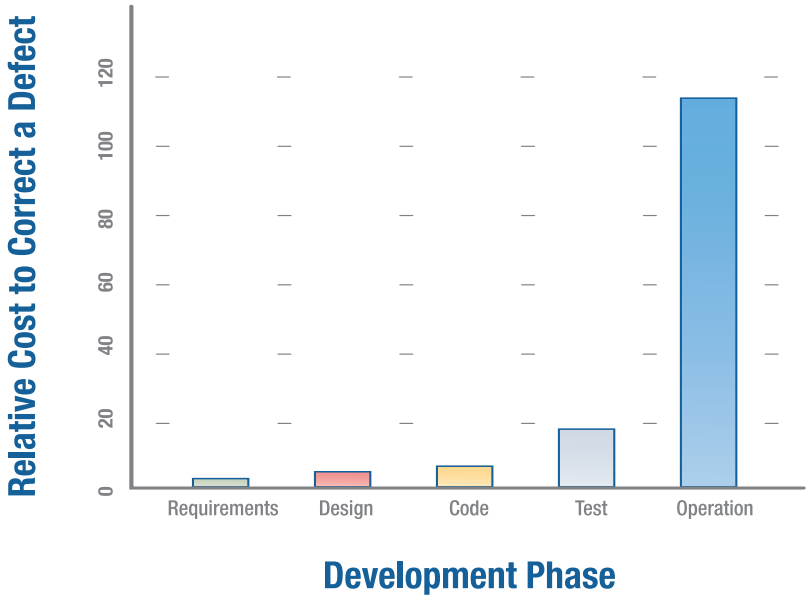

Figure 1: Relative cost to correct a requirement defect depending on when it is discovered.

Shortcomings in requirements cause no end of headaches on software projects. For example, creeping requirements contribute to schedule and budget overruns. Now, change is going to happen. Requirements change because initial elicitation activities are imperfect, because business needs evolve, and because customer expectations change once they see the product beginning to take shape. Project plans and commitments therefore need to anticipate change, rather than expecting that an initial stage of gathering requirements is all that's needed. Requirements creep, though, is the uncontrolled growth of requirements well beyond the original scope boundary.

The way in which requirements changes are incorporated into the product can degrade quality. You need to introduce changes at the highest level of abstraction they affect and follow the links in the traceability chain to see the impact of each proposed change. Traceability data stored in a requirements management tool such as Serena® RTM™ greatly facilitates this impact analysis.

One of the biggest problems with requirements is ambiguity. Ambiguity results when a requirement can be interpreted in multiple ways. Tools and activities that let stakeholders compare their understanding of the requirements can reveal these ambiguities. Ambiguity leads to wasted effort, as developers or testers interpret the requirement to mean something different from what the customer or analyst intended.

Many projects suffer from gold-plating. Developers might add unnecessary features they think the users will like, or users request excessively complex systems. Tracing each functional requirement back to its origin—such as a user task, a business rule, or a business objective—helps solve this problem.

The result of all of these kinds of requirements problems is to slow the pace of software development. It doesn't do you any good to work very efficiently on the wrong requirements. If you don't get the requirements right, it doesn't matter how well your team executes the rest of the project.

How Good Requirements Accelerate Development

A sign in a high-school chemistry lab reads, "If you don't have time to do it right, when will you have time to do it over?" We rarely take the time to understand a product's requirements clearly enough to reduce the risk of racing into construction. However, we *always* find the time, money, and resources needed to fix a flawed product. It seems counterintuitive: If you spend more time on requirements engineering in the early stages of a project, it's supposed to save you time later in the project? Absolutely! To convince yourself of the benefits, think about how much effort on your projects is misspent because of miscommunications, misunderstandings about requirements, missing information, or misstated requirements. The case for emphasizing solid requirements practices is an **economic argument,** not a philosophical or technical argument.

A solid foundation of agreed-upon business requirements and project objectives establishes a shared vision, goals, and expectations. Frequent and intimate user involvement makes software development a collaborative partnership between IT and the customer community. It also reduces the chance of system rejection upon rollout. It's frustrating for everyone to release a product only to find that essential functionality is missing and other capabilities are included that no one will ever use. Good requirements processes ensure that the functionality built enables users to perform essential business tasks. Well-specified requirements also define achievable quality expectations, letting the team implement both the capabilities and the characteristics that will make the users happy.

While typical projects spend perhaps 10 percent of their effort on requirements, investing more has a big payoff. A study of 15 projects in the telecommunications and banking industries revealed that the most successful projects spent 28 percent of their resources on requirements engineering (Hofmann and Lehner 2001). The average project devoted 15.7 percent of its effort and 38.6 percent of its schedule to requirements engineering. NASA projects that invested more than 10 percent of their total resources on requirements development had substantially lower cost and schedule overruns than projects that devoted less effort to requirements (Hooks and Farry 2001). And in a European study, teams that developed products more quickly devoted more of their schedule and effort to requirements than did slower teams, as shown in Table 1 (Blackburn, Scudder, and Van Wassenhove 1996).

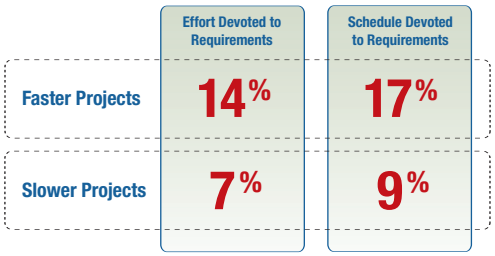

Table 1. Investing in Requirements Accelerates Development

Not all of the requirements development effort should be allocated to the beginning of the project. Don't think of it as a "requirements phase," but rather a set of requirements-development activities that are threaded throughout the project's lifecycle. Projects that follow an incremental lifecycle will work on requirements on every iteration through the development process. Agile development projects take an incremental approach, rapidly building small portions of the product so users can refine their needs and developers can keep up with changing business demands. Agile projects will have frequent but small requirements development efforts. Regardless of how large an increment your team tackles, they need to understand the requirements for that increment.

Managing Your Requirements

The goal of requirements *development* is to deduce, capture, and agree upon a set of functional requirements and product characteristics that will achieve the stated business objectives. Requirements *management,* then, involves everything that happens with the requirements in the product lifecycle once the requirements are developed, as shown in Figure 2. The demarcation between requirements development and requirements management is the defining of a requirements baseline. A baseline is a snapshot in time that represents the agreed-upon set of requirements for a specific product release.

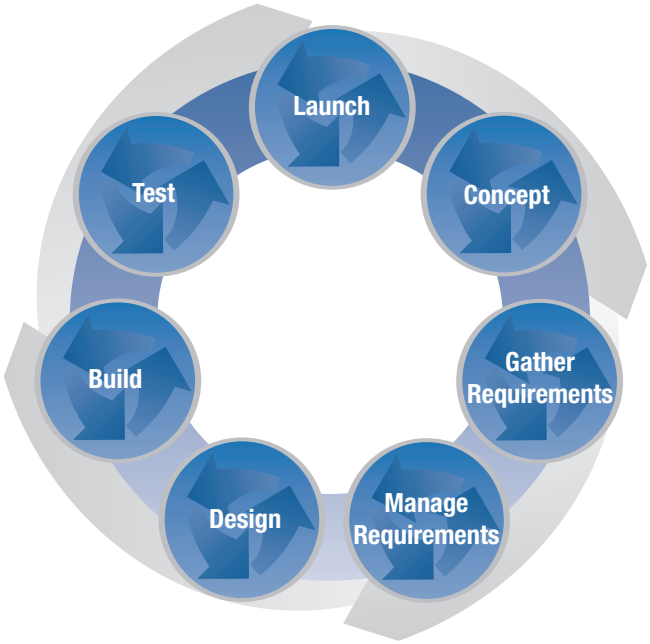

Figure 2: Product Lifecycle

The heart of requirements management is dealing with requirements changes. Each proposed change needs to be reviewed and the likely impact of implementing the change evaluated before it is approved by the appropriate body of decision-makers. This body is typically called the change control board. Change request decisions must be communicated to all who are affected so approved requirements changes can be incorporated into the software in a controlled way. Project plans need to be updated to remain current with the new requirements. The part of requirements management that no one likes – but which is the harsh reality – is the need to negotiate new commitments based on the estimated impact of changed requirements.

How Serena RTM Can Help

The traditional requirements approach is to create documents that contain business, user, functional, and nonfunctional requirements written in natural language, preferably using the organization's standard templates. A document-based approach to storing requirements has numerous limitations:

- It's difficult to keep the documents current and synchronized.
- Communicating changes to all affected team members is a manual process.
- It's not easy to store supplementary information (attributes) about each requirement.
- It's hard to define logical links between requirements of various types and other system elements.
- Tracking the status of individual requirements is cumbersome.
- Concurrently managing sets of requirements that are planned for different releases or for related products is difficult. When a requirement is deferred from one release to another, an analyst needs to move it from one software requirements specification (SRS) to the other.
- There's no convenient place to store those requirements that were proposed but rejected, nor those requirements that were deleted from a baseline. Sometimes you want to remember that these requirements had been proposed, because they might be back in scope one day.
- Reusing a requirement means that the analyst must copy the text from the original SRS into the SRS for each other system or product where the requirement is to be used.
- It's difficult for multiple project participants to modify the requirements, particularly if they are geographically separated.

A requirements management tool such as Serena RTM that stores information in a multi-user database provides a robust solution to these restrictions. Serena RTM lets users create various classes of requirements information and define unique sets of attribute values for each requirement class. Users can import requirements from source documents, filter and display the database contents, and export requirements in various formats. Users can define traceability links between pairs of objects stored in the database, as well as connecting requirements to items stored in other software development tools.

Serena RTM becomes even more valuable as time passes and the team's memory of the requirements details fades. Following are some of the tasks Serena RTM can help you perform.

- **Track requirements status.** Collecting the requirements into a database lets you know how many discrete requirements you've specified for the product. Tracking the status of each requirement during development supports the overall status tracking of the project. A project manager has good insight into project status if he or she knows that 55 percent of the requirements committed to the next release have been verified, 28 percent have been implemented but not verified, and 17 percent are not yet fully implemented.
- **Communicate with stakeholders.** Serena RTM permits team members to discuss requirements issues electronically through threaded conversations. Additionally, email messages can automatically be triggered to notify affected individuals when a specific requirement is modified or a change is proposed. Making the requirements accessible online can save travel costs and reduce document proliferation.
- **Store requirements attributes.** Possible attributes that provide a richer understanding of each requirement include priority, status, verification method, origin, person responsible, planned release, and so on. Everyone working on the project must be able to view the attributes and selected individuals be permitted to update their values. Serena RTM generates system-defined attributes such as the date a requirement was created and its current version number, and it lets you define additional attributes of various data types. You can sort, filter, or query the database to display subsets of the requirements that have specific attribute values. For example, you might ask to see a list of all the requirements planned for release 2.3 that were assigned to Julie and have a current status of Implemented.

- **Manage versions and changes.** Your project should define a requirements baseline. Serena RTM provides flexible baselining functions and maintains a history of the changes made to every requirement. You can record the rationale behind each change decision and revert to a previous version of a requirement if necessary. Serena RTM also contains a built-in change-proposal system that links change requests directly to the affected requirements.
- **Facilitate impact analysis.** Serena RTM enables requirements tracing by letting you define links between different kinds of requirements, between requirements in different subsystems, and between individual requirements and related system components—for example, designs, code, tests, and user documentation. These links then help you analyze the impact that a proposed change will have on a specific requirement by identifying other system elements the change might affect. It's also a good idea to trace each functional requirement back to its origin or parent so that you know where every requirement came from.
- **Control access.** You can define access permissions for individuals or groups of users. Serena RTM lets you share information with a geographically dispersed team through a Web interface to the database. The database uses requirement-level locking to permit multiple users to update the database contents concurrently.
- **Reuse requirements.** Storing requirements in the RTM database facilitates reusing them in multiple projects or subprojects. Requirements that logically fit into multiple parts of the product description can be stored once and referenced when necessary to avoid duplicating requirements.

Serena RTM is a powerful complement to a robust requirements engineering process. The tool alone won't guarantee that you write high-quality requirements, talk with the right user representatives or set the right priorities. However, Serena RTM can keep your project team aligned on the specified requirements for each release, so you all agree when you're done and there won't be so many surprises when you get there.

About the Authors

Karl E. Wiegers is Principal Consultant with Process Impact in Portland, Oregon. Prior to starting Process Impact in 1997, Karl spent 18 years at Eastman Kodak Company. His responsibilities there included experience as a photographic research scientist, software applications developer, software manager, and software process and quality improvement leader. Karl has led process improvement activities in small application development groups, Kodak's Internet development group, and a division of 500 software engineers developing embedded and host-based digital imaging software products.

Karl is the author of the books *Software Requirements, 2nd Edition* (Microsoft Press, 2003),*Peer Reviews in Software: A Practical Guide* (Addison-Wesley, 2002), and *Creating a Software Engineering Culture* (Dorset House, 1996). Karl has also written more than 160 articles on many aspects of software, chemistry, and military history. He has served on the Editorial Board of *IEEE Software* and as a Contributing Editor for *Software Development* magazine.

Karl received a B.S. degree in chemistry from Boise State College, and M.S. and Ph.D. degrees in organic chemistry from the University of Illinois. He is a member of the IEEE, the IEEE Computer Society, and the Association for Computing Machinery. He is a frequent speaker at software conferences and professional society meetings.

Sandra McKinsey is a Senior Product Marketing Manager for Serena Software. In her role, she is responsible for the Requirements & Traceability Management strategy and product development. With more than 15 years of experience in information technology, Sandra's background includes application development, managing development groups and managing technology products through their entire lifecycle.

Before joining Serena, Sandra served as a Senior Product Marketing Manager at Stellent and Agilent Technologies. Sandra holds a Masters of Business in Marketing from the University of Colorado and a Bachelor of Science in Information Systems from the University of Colorado.

References

Davis 1993: Davis, Alan M. 1993. *Software Requirements: Objects, Functions, and States.* Englewood Cliffs, NJ: PTR Prentice Hall.

Boehm and Papaccio 1988: Boehm, Barry W., and Philip N. Papaccio. 1988. "Understanding and Controlling Software Costs." *IEEE Transactions on Software Engineering* 14(10):1462–1476.

Leffingwell 1997: Leffingwell, Dean. 1997. "Calculating the Return on Investment from More Effective Requirements Management." *American Programmer* 10(4):13–16.

Grady 1999: Grady, Robert B. 1999. "An Economic Release Decision Model: Insights into Software Project Management." *In Proceedings of the Applications of Software Measurement Conference,* 227–239. Orange Park, FL: Software Quality Engineering.

Hofmann and Lehner 2001: Hofmann, Hubert F., and Franz Lehner. 2001. "Requirements Engineering as a Success Factor in Software Projects." *IEEE Software* 18(4):58-66.

Hooks and Farry 2001: Hooks, Ivy F., and Kristin A. Farry. 2001. *Customer-Centered Products: Creating Successful Products Through Smart Requirements Management.* New York: AMACOM.

Blackburn, Scudder, and Van Wassenhove 1996: Blackburn, Joseph D., Gary D. Scudder, and Luk N. Van Wassenhove. 1996. "Improving Speed and Productivity of Software Development: A Global Survey of Software Developers." *IEEE Transactions on Software Engineering* 22(12):875-885.

About Serena

Serena Software is the largest company solely focused on managing change in the IT environment. Serena's products and services automate process and control change for teams managing development, web content and infrastructure. Based on 25 years of innovation in process and configuration management, Serena Application Framework for Enterprises (SAFE™) solutions enable customers at more than 15,000 sites worldwide and 98 of the Fortune 100 to improve IT governance, mitigate risks, support regulatory compliance, and boost productivity and quality. Serena is headquartered in San Mateo, California, with offices throughout the U.S., Europe, and Asia Pacific.

serena™

Yphise APPROVED
Serena RTM
Nb 050401
valid until 04/30/06