Jama software®

EBOOK

Jama Software's Guide to Requirements & Requirements Management

Table of Contents

CHAPTER ONE

Introduction to Requirements

Requirements are capabilities that a product must meet to satisfy a stakeholder's need to solve a specific problem. The stakeholder's needs can come from a number of sources, including compliance to a standard or regulation, a business need, a technical situation, market need, competition, etc.

If we consider the definition of the discipline of Requirements Engineering (i.e., all things associated with requirements management), it's this:

> Requirements Engineering (RE) refers to the process of defining, documenting, and maintaining requirements in the engineering design process."

Requirements are the foundation of a smooth-running process and are the essential inputs to your mission-critical projects. Effectively managing the flow of changes and refinements early in your lifecycle will significantly reduce both quality issues downstream and the volatility that plagues so many projects. Mapping your existing review and standards workflow onto a continuous, centralized, and collaborative process will enable you to maintain quality standards, meet goals, and reduce your overall delivery time.

A recent study showed that of all failed projects investigated, more than 40% concluded that failure was because of bad requirements or being unable to understand the stakeholder's true need in the first place.

Without good engineering practices to discover and formulate correct and consistent requirements and making that specification complete (to describe the full need of functions, qualities, and constraints), you will still have huge problems, whether you have a modern requirements management solution or not. Garbage in is, and will always be, garbage out.

CHAPTER ONE | SECTION ONE

The Business Value of Better Requirements Management

Not every manager is convinced that their team needs to do a better job on requirements development and requirements management, or that such an investment will pay off. Numerous industry studies, however, indicate that requirements issues are a pervasive cause of project distress. The often-quoted CHAOS Reports from The Standish Group indicate that three of the biggest contributors to projects that fail or are "challenged" are lack of user input, incomplete requirements and specifications, and changing requirements and specifications.

An Economic Argument

The case for implementing better requirements practices is an economic or business argument, not a philosophical or technical position. Think about how your company's bottom line is affected by requirements-related problems. Then use that understanding to justify investing in better requirements practices that can pay off for the long term.

The main reason errors in requirements are so damaging is that they force the development team to perform extensive rework to correct the errors. It's well established that the cost of correcting a software error increases dramatically the later it is discovered, as shown in **Table 1**. An error, omission, or misunderstanding in the requirements forces developers to redo all the work they've already done based on the incorrect requirement. Therefore, any technique that can reduce requirement defects and prevent some of this wasted effort is a high-leverage investment indeed. One analysis of the potential return on investment from better requirements suggested that requirement errors can consume between 70 and 85 percent of all project rework costs.

TABLE 01

Relative Cost to Correct a Requirement Error

STAGE ERROR IS DISCOVERED	RELATIVE COST TO CORRECT
Requirements Development	1x
Design	2–3x
Construction	5–10x
System or Acceptance Test	8–20x
Operation	68–110x

CHAPTER ONE | SECTION TWO

What Better Requirements Can Do for You

In addition to avoiding some of the negative consequences described so far, better software requirements provide numerous benefits. These include selecting the right projects to fund, facilitating estimation, enabling rational prioritization, developing higher quality designs, and testing more effectively.

Selecting Projects to Fund

Good preliminary requirements enable senior managers to make effective business decisions as organizations decide which among a set of potential projects to fund. Better requirements allow a more accurate projection of business returns. Once a project is funded, better requirements allow project managers to more sensibly partition tasks among their teams and even among individual team members.

Facilitating Estimation

Well-understood requirements can help your team estimate the effort and resources needed to execute a project. Reliable estimation requires some historical correlation between requirements size and effort.

Enabling Prioritization

Documented requirements allow the team to prioritize its remaining work. Most projects need to make compromises to ensure that they implement the most critical and most timely functionality. A prioritized requirements baseline helps the team incorporate those changes that will deliver the maximum customer value. One study revealed that just 54 percent of the originally defined features were delivered in an average project. If you can't implement all of the requested functionality, make sure the team implements the right portion.

Developing Designs

Requirements are the foundation for design. Therefore, well-understood and well-communicated requirements help developers devise the most appropriate solution to the problem. High-quality requirements also ensure that the development team works on the right problem. Many developers have experienced the frustration of implementing functionality that someone swore they needed, only to find that no one ever used it. One survey by The Standish Group indicated that fully 45 percent of the delivered software product features were never used. Wasting less time implementing the wrong functionality accelerates the project and maximizes its business return.

Testing Effectively

Well-defined and testable requirements allow testers to develop accurate test procedures to verify the functionality. Prioritizing requirements tells testers which ones to concentrate on first. Assessing requirement difficulty and risk helps testers know which functionality should receive the closest scrutiny.

Testing the Requirements

Testing requirements early and often reduces development time and helps control project costs.

READ THE PAPER >

Tracking Project Status

A comprehensive, traced set of requirements helps the stakeholders know when the project is done. A body of work is complete when all of the requirements allocated to it are either verified as being correctly implemented in the product or deleted from the baseline. Defined business requirements also allow the stakeholders to determine whether the project has met its goals.

Accelerating Development

Believe it or not, investing more effort in developing the requirements can actually accelerate software development. This seems counterintuitive, but it's true. Defining business requirements—the expected business outcomes the product will provide—aligns the stakeholders with shared vision, goals, and expectations. Effective user involvement in establishing the requirements reduces the chance that users will reject the new system upon delivery.

Accurate requirements ensure that the functionality built will let users perform their essential business tasks. The requirements also establish achievable quality expectations. This lets the team implement both the capabilities and the product characteristics—the nonfunctional requirements—that will make users happy. Additionally, emphasizing requirements development is cheaper than relying on beta testing to find requirements problems. Fixing problems that late in the game is far costlier than correcting them earlier.

TABLE 02

Cost and Schedule Overruns on Some NASA Projects a Requirement Error

PERCENT OF BUDGET SPENT ON REQUIREMENTS	NUMBER OF PROJECTS	AVERAGE PROJECT COST OVERRUN
< 5%	5	125%
5 to 10%	7	83%
> 10%	6	30%

CHAPTER TWO

The Fundamentals of Effective Requirements Management

Improving your requirements management process can have major impacts on your development process. Some of the benefits include improving your efficiency, speeding time to market, and saving valuable budget and resources.

Requirements are the information that best communicates to an engineer what to build, and to a quality-assurance manager what to test.

A requirement has three functions:

- Defines what you are planning to create
- Identifies what a product needs to do and what it should look like
- Describes the product's functionality and value

Requirements vary in complexity. They can be rough ideas sketched on a whiteboard or structured "shall" statements. They can be text, detailed mock-ups, or models, and can be part of a hierarchy with high-level requirements broken down into sub-requirements. They may also be detailed specifications that include a set of functional requirements describing the behavior or components of a product.

High-level requirements are sometimes referred to simply as "needs" or "goals." Software development practices might refer to requirements as "use cases," "features" or "functional requirements." Agile development methodologies often capture requirements as "epics" and "stories." Regardless of the terminology, requirements are essential to the development of all products. Without clearly defining requirements, companies risk creating incomplete or defective products.

Throughout the process, there can be many people involved in defining requirements. A stakeholder might request a feature that describes how the product will provide value in solving a problem. A designer might define a requirement based on how the final product should look or perform from a usability or user interface standpoint. A business analyst might create a system requirement that adheres to specific technical or organizational constraints.

CHAPTER TWO | SECTION ONE

Four Fundamentals of Requirements Management

Today's sophisticated products and software applications often take hundreds or thousands of requirements to sufficiently define the scope of a project or a release. Teams must be able to access, collaborate, update, and test each requirement through to completion because requirements naturally change and evolve over time during the development process. They must all understand the four fundamentals of requirements management:

Design Teams: Requirements Management & Product Complexity

A recent study by Engineering.com ties effective requirements management to better product outcomes.

READ THE REPORT >

01

Good Requirements

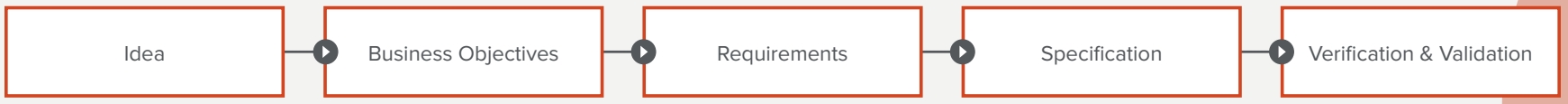
A good requirement should be valuable and actionable. It should provide a pathway to a solution, and everyone on the team should understand what it means. Good requirements need to be concise and specific, and should answer the question "What do we need?" rather than "How do we fulfill a need?" With accurate requirements, stakeholders can understand their part of the plan. If they lack this knowledge, and if requirements are unclear or vague, the final product could be defective or fail.

02

Collaboration and Buy-In

It's difficult to get a company to agree on requirements, particularly for large projects with many stakeholders. In practice, it's not necessary to achieve consensus through compromise. It's more important to have team buy-in so the development process can move forward. With buy-in, the team backs the best solution, makes a smart decision and does what is necessary to move forward.

Team collaboration is key to establishing good requirements. Collaborative teams work hard to make sure everyone has a stake in the project and provides feedback. When there is a commitment and understanding of project goals, team members tend to support others' decisions. It's when developers, testers, or other stakeholders feel "out of the loop" that communication issues arise, people get frustrated, and projects get delayed.



03

Traceability and Change Management

Requirements traceability is a way to keep everyone in the loop. It organizes, documents, and keeps track of all requirements, from initial idea to testing. A simple metaphor for traceability is connecting the dots to identify the relationships between items within a project. The following figure shows an example of a common downstream flow. Companies should be able to trace each requirement back to its original business objective throughout the development process, not merely after it's complete. By tracing requirements, companies can identify the ripple effect changes have, see if they have completed a requirement, and if they tested it properly. With traceability, and by managing changes effectively, managers gain the visibility needed to anticipate issues and ensure continuous quality.

Traceability also makes sure the product meets all the vital requirements that come from different stakeholders. By tracing requirements, all team members stay connected to each other and to all interdependencies. And by managing changes well, a company can avoid scope creep — unplanned changes that occur when requirements are not clearly captured, understood, and communicated. The benefit of good requirements is a clear understanding of the product and the scope involved. This leads to a better development schedule and budget, which prevents delays and cost overruns.

Better Product Development: Five Tips for Traceability

Get more expert advice on leveling up your traceability process.

READ THE PAPER >

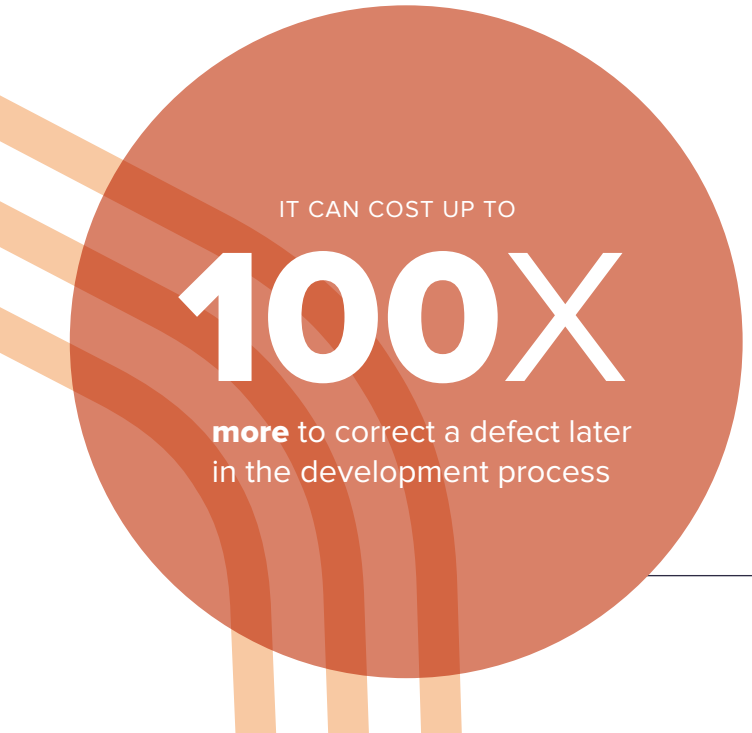
04

Quality Assurance

Getting requirements right the first time means better quality, faster development cycles, and higher customer satisfaction with the product. Concise, specific requirements can help companies detect and solve problems earlier rather than later, when they are much more expensive to fix.

Research has shown that project teams can eliminate 75 to 80 percent of project defects by effectively managing requirements. In addition, according to Borland Software (now Micro Focus), it can cost up to 100 times more to correct a defect later in the development process, after it's been coded, than when it's still in written form.

By integrating requirements management into their quality assurance process, companies can help teams increase efficiency and eliminate rework. According to the Carnegie Mellon Software Engineering Institute, 60 to 80 percent of the cost of software development is in rework. In other words, development teams are wasting the majority of their budgets on efforts they didn't perform correctly the first time.



CHAPTER THREE

Figuring Out the Right Level of Detail in Requirements

In today's hyper-competitive marketplace, companies are releasing products faster than ever before. However, that increased emphasis on time to market doesn't change the expectations: producing quality products while, when necessary, conforming to all relevant safety and quality standards.

Finding that critical balance between speed and quality is essential for product teams, which is where requirements management comes into play. Unfortunately, a lot of teams don't take the time to flesh out requirement details – or they have the opposite problem and provide too many details and confuse requirements with design specifications. Here are some examples and recommendations to make sure you're getting the most value from your requirements.

SCENARIO 01

I Need More Details!

In this situation, teams fall into a trap of not taking the time to define requirements. For example, a marketing team — which tends to think more strategically — will often give high-level problem statements directly to an engineering team, which typically thrives on details.

What happens next is either:

1. Engineering team builds a product that doesn't solve the high-level problem.

   OR

2. Engineering team will bombard the marketing team with requests for additional information until both sides are frustrated and late on delivery.

Neither scenario results in a smooth development process, and both can breed mistrust between teams whose ultimate goal is the same, in spite of their differing roles.

Well-formed and reviewed requirements clear the path for a quality product that meets market needs and expectations. They ensure each team member is on the same page, and that everyone involved knows exactly what they're building and why.

SCENARIO 02

Too Many Details – I Can't Innovate!

In this case, the team members responsible for authoring requirements can sometimes cram too many design details into the requirements themselves. For example, "the button shall be red and shall be located in the top-right corner of the console."

First off, that's actually two requirement statements – but either way – it's describing implementation detail and, taken on its own, I have no idea of what the button should do or why. Is this supposed to dispense a soda from a vending machine or fire a missile from a battleship?

Requirements like these create friction within engineering teams, limiting their ability to innovate while getting bogged down with overly specific instructions.

Engineering teams should ideally be free to iterate and innovate the design rather than simply doing what they're told. The requirements should focus on the need and keep teams aligned on what is needed and why while engineering experts figure out how to implement.

RECOMMENDATION

Establish a Common Taxonomy and Trace Model

An important step in striking the right level of detail in your requirements is establishing common terms in your product development process.

Many teams use different terms to refer to the same thing (e.g., "specification" – this term means so many different things across different teams).

Having a glossary or standard that clearly delineates important terms and deliverables facilitates clarity throughout the development process and reduces the confusion referenced above.

Using a traceability model establishes appropriate hierarchy and helps break down high-level problems to detailed requirement needs. It also can facilitate test coverage of requirements. This clarity is especially important when dealing with life-critical systems, such as medical devices or autonomous vehicle systems. In these cases, missing requirements or inadequate design and testing can quite literally cost lives.

Finally, teams should be able to differentiate requirements from design, the "need" versus the "how." Bundling both together inside requirements can lead to added complexity, unnecessarily constrained design, and friction between teams.

Quickly bringing a quality, safe product to market is a challenge. Requirements and design are both key elements to doing so. Making sure teams strike the right level of detail in the requirements is critical in avoiding confusion and reducing friction in the process.

Conclusion

Requirements management can appear to be a complex topic, but at its core, it's a simple concept. It helps teams answer the question: Does everyone — from business leaders to product managers and project leaders to developers, QA managers and testers — understand what is being built and why?

When everyone is collaborating seamlessly with full context and visibility into the discussions, decisions, and changes that occur throughout the development cycle, there's a better shot at maintaining a high level of quality and ensuring success.