Requirements Engineering Best Practices

July 2015

by Kevin Parker, Vice President of Worldwide Marketing, Serena Software (Now Part of Micro Focus®)
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Are Requirements Still “a Thing”?</td>
<td>1</td>
</tr>
<tr>
<td>Requirements Is a Process</td>
<td>2</td>
</tr>
<tr>
<td>Best Practices for Managing Requirements</td>
<td>5</td>
</tr>
<tr>
<td>Critical Attributes of a Requirements Management Solution</td>
<td>6</td>
</tr>
<tr>
<td>Summary</td>
<td>13</td>
</tr>
</tbody>
</table>
Are Requirements Still “a Thing”?

A requirement defines one, and one only, atomic behavior that is needed or wanted by the business in order to achieve its goals. The requirement must be testable in measurable ways. It must be complete, containing all necessary information. It must not contradict any other requirement and must be consistent with legal standards. It must have at least one stakeholder, at least one sponsor, and benefit at least one user.

A Requirement Defines a Property That Is Essential for the Business

The definition is straightforward. The range of methodologies adopted for gathering and managing requirements varies more than any other Application Lifecycle Management (ALM) discipline. Some differences result from the level of specificity needed for a given project due to product scope, application complexity, or team size and distribution. Within a single organization, one team may need requirement perfection with detailed review by stakeholders, while another team may start with a simple idea and iterate and prototype until there is just enough understanding to commence building the solution.

The IEEE defines requirements as “… the conditions or capabilities that a system (or system component) must meet to satisfy a contract, standard, specification, or other formally imposed constraint …”

Karl Wiegers¹, the foremost authority on requirements engineering, simplifies the definition to embody the properties that a product must have in order to provide value to a stakeholder. All work in the software development lifecycle (SDLC)² begins with an idea. That idea might be a new system to be developed, a request for an improvement to an existing system, or it could be a need to fix something that isn’t behaving as it should.

We call these new systems, enhancements, and fixes. We categorize, prioritize, and organize these into releases, versions, and patches.

- **New**—New systems or new features of an existing system, these are usually the larger projects in an organization, and that makes them the most expensive. Funding for these projects is, consequently, harder to get. Estimating, planning, and execution are more difficult because we are dealing with many unknowns.

  Example: Add the capability to show the product catalog on a mobile device.

---


Enhancements—Improvements to existing code, this is incremental work and the costs are lower, the risk is less, and funding is generally easier to obtain. These can be short or long projects. They are easier to estimate and cost because they are a known problem and solution set.

Example: Make the product catalog display the prices in Euros, Pounds, and Yen in addition to U.S. dollars.

Fixes—Often small changes to existing code that do not add new functionality, fixes change the behavior of current functionality to meet the expectations of the business. Most fixes address behaviors that are wrong and damaging to the business, but sometimes they modify behavior so that it just works better or faster.

Example: On checkout, tax is added on domestic sales only and not on all sales worldwide.

Taking the idea and turning it into a solution is what ALM is all about. Getting there is quite a journey. In fact, just getting the idea captured in the first place is no small task.

Requirements Is a Process

Defining the problem can takes minutes or months. It depends on many factors. Here are just some of the things the organization takes into account when defining a problem:

- Regulatory and compliance controls
- Security and integrity of the data
- Access and authentication
- Time-to-market
- Availability of commercial solutions
- Technical complexity
- Return on investment
- Disruption to project plans
- Availability of resources
- Funding
- Performance needs
- Internationalization needs

All this before we ask, “What exactly do you want this technology to do?”

---

Ten Best Practices for Eliciting Great Requirements:

1. Use prototypes and simulation instead of written definitions.
2. Share ideas with as many stakeholders as possible and garner feedback.
3. Use gamification techniques to get balanced feedback.
4. Every requirement should be testable.
5. Every requirement should be about one thing.
6. Maintain traceability from request to requirement to implementation.
7. Use story boards, use cases, pseudo code, or whatever gives clear meaning.
8. Test all requirements for their impact on existing and future features.
10. Version requirements and document changes.
Step 1—Ideation and Demand Capture: It is common for several ticketing systems to be in use simultaneously for different kinds of requests and for different systems. Some systems are self-help and others require an email or phone call. All too often, Microsoft Excel is the repository for these requests. Most organizations have a way of collecting ideas from the business, usually done via some kind of “ticketing” system, often called a Change Request System (or CR System). These are typically hosted and managed by the Help Desk (also known as the Service Desk).

An essential element of this process is the collection of ideas in a consistent and traceable manner. Ideation is a more modern concept and not formalized in many organizations. In some enterprises, this is the Change Control Review Board (CCRB) or the Change Advisory Board (CAB) and many other similar titles. All too often, MS Excel is the system of record and this makes collaboration and visibility very difficult.

Step 2—Change Management: This is where triage of the ideas happens so that the most important ones, the time-critical ones, and the governance-critical ones get the highest priority. The ideas go through a feasibility research, funding, resourcing, and scheduling process before approval for development. Some urgent and emergency requests get approval immediately when the business impact is severe.

Sometimes called demand management, this is a more recent phenomenon being driven by IT, which is more accountable for the money spent. The purpose of demand management is to ensure that the projects that IT works on are the projects that match the business priorities. Frequently, CIOs will talk about “alignment” or “aligning IT with the business”: when they do, they are describing the need to ensure that mission critical, compliance essential projects are the ones implemented first and fastest. The review board may oversee this process and set priorities, goals, and even budgets.

Step 3—Requirements Elicitation: This involves the gathering and recording of requirements from stakeholders and other sources. There are several techniques used, including interviews; document analysis; focus groups; workshops; and more recently, prototyping, simulation, and gamification. Iterative prototyping defines requirements in less time than it takes to interview and re-interview the business user.
Step 4—Requirements Validation: This confirms that the set of requirements meet the business needs and goals. This step exists to ensure that the information gathered is sufficient to build the desired solution, and that the information is in conformance with organizational constraints such as compliance and governance standards.

Frequently, the definition of the requirements fails because the English language is too imprecise to reflect the nuanced details of the business need. Development of use cases³ and the use of formalized methods such as Unified Modelling Language (UML)⁴ help define with precision exactly what the expected behaviors will be for the solution. These tools assist in the development of test cases that will form the basis of validating the implementation against the requirement later in the development lifecycle.

Step 5—Requirements Management: Often, this activity follows very formal procedures, and it is the process where requirements are improved, versioned, tracked, monitored, prioritized, assigned, and, in short, managed. These activities include:

- The continued refinement of requirements as more data, input, and ideas are gathered
- The ordering and prioritization of requirements depending on business input
- The scoping of individual requirements into effort
- The organizing of requirements into groups that make reasonable pieces of work and releases to be completed by the development teams
- The tracking of changes, updates, and modifications of requirements as business needs are refined
- The maintenance of traceability to ensure an audit trail from request to implementation
- The monitoring of approvals for requirements to be implemented for the funding of projects
- The maintenance of the relationships and dependencies between and amongst requirements

The process of managing requirements is essential to application development best practices. Without knowledge of what one is supposed to develop, the likelihood that it will match the business need is almost zero. No matter how the term is defined, or where on the methodological spectrum your organization finds itself, the expectations of the customer (or the potential customer) must be understood before construction begins. The “how” may be worked out in development, but the “what” must be understood before the funding is approved.

In an Agile organization, the process followed is much the same except that there are many more iterations in the time that a more traditional requirements engineering activity would take.

The key person on an Agile team is the Product Owner, whose job is to act as the liaison between the business and the development team.

The Product Owner gathers the requirements and orders and prioritizes them for the development team. The Product Owner performs triage with the business, helping the business stay focused on asking for what is most important.

Agile teams don’t prototype. They get to working code as quickly as they can.

Projects are managed through the Epics (a collection of requirements) and Stories (a requirement) that make up the Sprint Backlog (requirements assigned to dev but not yet completed) of work to be done.

Progress is tracked through Burndown Charts (requirements completed) showing progress towards the end of the current Sprint. A key principle of Agile is to be self-organizing, and the Daily Standup meeting keeps everyone synchronized within the project.

The Project Backlog is constantly reprioritized to meet ever-changing business needs.

---

### Best Practices for Managing Requirements

**Expect requirements churn to be 1% to 5% per month.**—source Gartner

“The best practice for requirements management is to apply the best practices defined for configuration management to the management of requirements.”—Kay Fuhrmann, Product Manager, Serena Software (now part of Micro Focus)

1. **Naming conventions.** Defining and maintaining conventions for identifying releases—from the approved requirement set through to the baselined release to the emergency fix or patch.
2. **Baseline requirements.** Requirements, like software releases, must be baselined and those baselines must map directly to the releases they produce.
3. **Well-defined and understood change control process.** Once a baseline is created, changes must be controlled, tracked, traced, approved, and reviewed.
4. **Requirements review.** There must be a requirements review process, and it must be enforced.
5. **Expectation of changes.** Make sure changes can be made easily but under strict access control rules with full traceability.
6. **Version management.** Requirement history should be maintained using methods that make it easy for analysts to look back.
7. **Requirements traceability.** Without the ability to trace a requirement from the idea through to its defined implementation, there is no ability to understand the impact of a proposed change.
8. **Information maintenance.** Maintain attributes for dependencies, relationships, owners, stakeholders, users, funder, dates, costs, models, prototypes, diagrams, governance, etc. about the requirement.
9. **Collaboration.** Provide easy access to requirements information and automatically notify stakeholders of any change of status or change of the requirement to foster collaboration.
10. **Requirements in a single location.** Keep requirements in a single location, preferably in a database designed to manage them.
Critical Attributes of a Requirements Management Solution

Ease of Use

Implementation
The solution implementation must be easily configured to adopt the language used within the organization. It must not be a requirement that the lexicon change to fit the solution. Training, reasonable implementation, and localization should be completed in two to four weeks.

Solution must be easy to learn
New users, no matter their role, can quickly learn to log into the solution and to navigate within the client interface. In this context, nontechnical requirement reviewers, with an understanding of the corporate requirements, can begin work with a one page overview. All other users of the solution, with the exception of administrators or project managers trained to support the implementation, must be in a position to use the product after two to four hours training.

Solution must be easy to use
The solution selected for the management and reporting of requirements must make the jobs of all involved easier while increasing their access to information. It must take no longer than one hour for the general user to understand how to perform specific functions using the solution.

Properties and display must be configurable
It must be possible for users or administrators to modify the amount of information displayed.

All stakeholders must have access to documents
It must be possible to publish documents in corporate format for review and approval. It should not be required that users use the tool in order to see the information contained within.

The tool philosophy and technology must be transparent
Users should not need any architectural, administrative, or technical ability in order to use the tool to add, modify, review, or approve requirements and their properties.

Top 10 Most Requested Features of a Requirements Management Solution—source Forrester
- Requirements baseline to track scope drift
- Requirements modeling and simulation
- Visual tools to manage requirements into releases, features, and patches
- Decision support to prioritize selection of requirements
- Linking and tracing relationships amongst requirements and requests
- User-centric requirements capture
- Requirements reuse of common and shared requirements
- Requirements workflow that parallels other SDLC workflows
- Integration based on traceability with other ALM lifecycle tools
- Out-of-the-box requirements for common compliance needs such as ITAR
User requirements (sometimes referred to as business requirements) identify capabilities that the stakeholder wants or needs, while functional requirements specify what the system must do in order to provide the stated capability.

There must be methods for organizing requirements
It must be possible to create containers (folders) as a means of organizing requirements according to component, subtype, or to create lists for specific groups or users.

User roles must be supported
It must be possible to define organizational or project roles and to map users to those roles.

The System Must Be Flexible
Requirements are distinguished by type or classification. Common requirement types include user and functional requirements. User requirements (sometimes referred to as business requirements) identify capabilities that the stakeholder wants or needs, while functional requirements specify what the system must do in order to provide the stated capability. Other common requirement types include non-functional, system, technical, and test cases. Beyond these basic types, the classifications used for requirements are as diverse as the software to be developed or the processes employed by the organization to do so. A good requirements management solution must be capable of providing a means for classifying any requirement the organization wants to define, and its properties (metadata) must also be fully configurable.

Requirements must be stored as individual objects
Requirements of all types must be stored as individual objects. They may be linked to but not owned by other objects in the project. Any combination of these objects may be gathered for display or publication.

Requirements must be able to be organized in hierarchies and decomposed to the atomic level
Requirements of all types must be able to be split into increasingly smaller requirements until they reach atomic size that represents one, and only one, specific requirement. Hierarchies of decomposed requirements must be able to be operated on individually, at the atomic level, and at the hierarchy and intermediate hierarchy level with action cascading down to the decomposed requirements.

Requirement types must reflect corporate standards
Requirement types must be named and described according to the organization’s needs and processes, rather than those of the tool provider. There must be no requirement within the solution to include any out-of-the-box requirement type.
Properties must be configurable
There must be the ability to define and assign properties to requirement types, beyond those defaults defined by the product. The data types used in the definition of these properties must be flexible as well. For example, they must include text fields (short and long), dropdown lists, numeric fields, and date fields. It must be possible to associate graphics or attachments with data types.

Properties need to use familiar terms
The language used to define requirements and their properties must reflect the lexicon of the organization, and must be referred to consistently across the solution. It must be possible for multiple project teams from within the same organization to define requirement types and to use terms that meet the needs of each. No group should be required to carry the baggage of others.

There must be traceability
It must be possible to create a traceability view, with mechanisms for selecting the requirement types to be traced. The information gathered must be displayed in a format that is easy to read and to analyze.

Importing Requirements
Most project teams, prior to adopting a requirements management solution, maintain requirements in MS Excel or MS Word documents. It is critical that the solution import existing data, and it must be capable of continuing to import requirements updates.

Import data from MS Word files
The solution must provide functionality to facilitate the import of data from MS Word files. It must be possible, for example, to import the requirements in this document.

Import data from MS Excel files
It must be possible to import selected data from MS Excel files in order to create new requirements, with simple mechanisms for mapping columns to properties.

Export into MS Excel files
It must be possible to export selected requirements data into MS Excel files, with simple mechanisms for mapping properties to columns.
Update existing objects from MS Excel

The solution must support the functionality to update existing requirements with data from MS Excel worksheets. Mapping requirement identifiers, or names from the MS Excel sheet to the current requirement and then adding new properties or updating existing properties will enable all stakeholders, even those without access to the solution, to take an active role in the review process. Updates from MS Excel worksheets will also ensure batch capabilities.

Establish or update links using worksheets

Links between requirements can be established using MS Excel worksheets.

Import data from associated ALM solutions

MS Excel or CSV files generated by any solution along the ALM spectrum can be imported in order to link, modify, add, or delete requirements or their properties.

Mechanisms for Filtering and Sorting Requirements

User controlled filters

Requirements are managed, viewed, and reviewed at different times in the development cycle and by different people with very different needs. It is seldom necessary for anyone to see everything.

Lists and reports filtered by requirement property

It must be possible to filter on any non-binary property defined within the requirement.

Establish and update requirement sets

It must be possible to create sets of requirements of any type and to establish baselines from those sets.

Queries to support requirement selection

The tool must support the use of simple or complex queries in order to collect requirements based on type, property, absence or presence on a list, relationship (link), or a combination of these; as an example, all high priority user requirements that are related to at least one test case of which the test result property is set to “failed.”
Simple creation of documents, lists, or csv files
It must be possible to save, print, or create MS Excel or CSV files from any dialog in the user interface.

The Solution Must Manage Requirement History
The solution must be capable of providing a mechanism for creating, deleting, and changing a requirement as well as any and all of the metadata stored with the requirement. There must also be the possibility of maintaining a record of all such transactions.

Requirement history must be maintained
The solution must be capable of maintaining the change history for all requirements and requirement properties.

Requirements can be deferred or abandoned
The solution must provide the functionality to abandon or defer requirements. It must be possible to easily filter these requirements from the general lists while maintaining accessibility for review or re-adoption.

Requirements can be deleted
The solution must allow a requirement to be permanently deleted from the project.

Baselines can be created and updated with agility
There must be facilities for the creation of release baselines that can be versioned and tracked for changes. It must be possible to compare baselines and to create a simple list of changed requirements, as well as a detailed report of the differences between baselines.

Requirement Linking and Traceability
Not only must a good requirement system be capable of providing a means of classifying any requirement type, but it must also be capable of linking requirements of various types. Requirement linking provides the mechanisms for traceability reports.
Requirement linking
There must be functionality to create relationships between requirement types. The links that express these relationships must be one to many, many to one, or many to many. For example, there can be multiple test cases for a single functional requirement, but one test case could be applicable for more than one functional requirement.

Manual links
The solution must provide the ability to establish links manually or through a form of batch import.

Deleted links
The functionality to delete links must exist.

Linked objects report impact of change
Linked objects must show the impact of upstream changes. For example, given a test case linked up to a functional requirement that is linked up to a user requirement, a change to the user requirement will assume possible impact and therefore force a review of the functional requirement and the test case. A change to the functional requirement will force a review of the test case, but not of the user requirement.

Traceability matrix
A complete traceability matrix using any defined requirement type must be available and relatively easy to use. Traceability visualizes the connections between requirements, enabling the stakeholders to see that user requirements have been detailed through associated functional requirements, and that test cases have been established for each functional requirement.

The traceability matrix must report fulfillment, and for the life of the application, must report the impact of change.

There Must Be an Excellent Reporting Functionality
The solution must facilitate good reporting, and it must be possible to generate documents using a format that is familiar to stakeholders. Maintaining up-to-date reports is critical to the process of requirements management, and creating such reports must be a simple extension of filtering and sorting from the client interface.
**Flexible document support**
It must be possible to select requirement lists using scripts or queries and to easily use those lists to create or to update a document.

**Version control documents**
It must be possible to freeze the contents of a document and to assign the frozen document a name or version. It must also be possible to delete such documents freely.

**Requirements publishable in document form**
It must be possible to export (publish) documents using a format (template) that is familiar, and that can be shared among projects.

**Facility to compare documents**
The solution must provide the ability to compare versioned documents.

**Graphical reporting**
It must be possible to create graphical reports or to gather and export the data needed to create reports current existing tools. Reports should include, but not be limited to, bar graphs showing scheduled requirements by type, number scheduled for a named release, and number yet to be implemented.
Summary

The discipline of requirements management is difficult for some organizations to get their hands around, and because of this, groups often overload the process to the detriment of the project. Project teams will often spend more time arguing about the proper use case structure, or the difference between a user and business requirement than about the content of either. Because such arguments do not move the project forward, team members will move on and just assume that open questions will be addressed during development.

The process of managing requirements is essential to application development best practices. Without knowledge of what one is supposed to develop, the likelihood that it will match the business need is almost zero. Requirement management can begin with a list on which project expectations are defined and managed. The list can be displayed like an open chalkboard, collecting and sorting notes and changes. It can also be expanded to control technical detail and to link the detail to test cases, ultimately allowing full requirements reporting capabilities.

The best requirements management solutions are easy to learn, easy to use, accessible, and transparent. They also have excellent traceability, versioning, and reporting functionality. These features work together to provide the foundation of a world class solution and the beginning of a best practice requirements process.