In Search of Excellent Requirements

Karl E. Wiegers

Process Impact

www.processimpact.com

Course Objectives

At the end of this course, you will be able to:

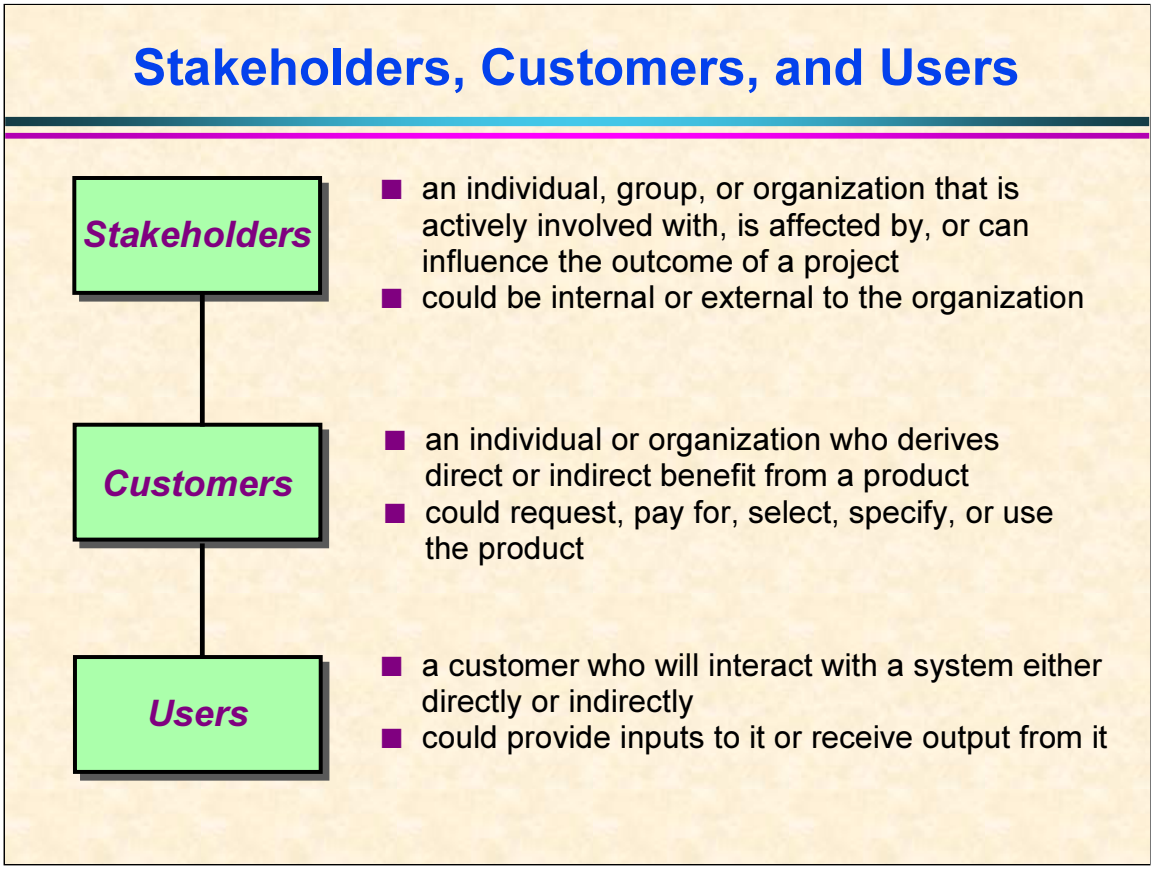
- Recognize different types of requirements information.
- List many good practices for requirements elicitation, analysis, specification, validation, and management.
- Select appropriate techniques for representing requirements on your projects.
- Critically evaluate requirements statements for ambiguities and other problems.
- Be prepared to manage requirements changes effectively on your projects.

Module 3: Customer Involvement

Learning Objectives:

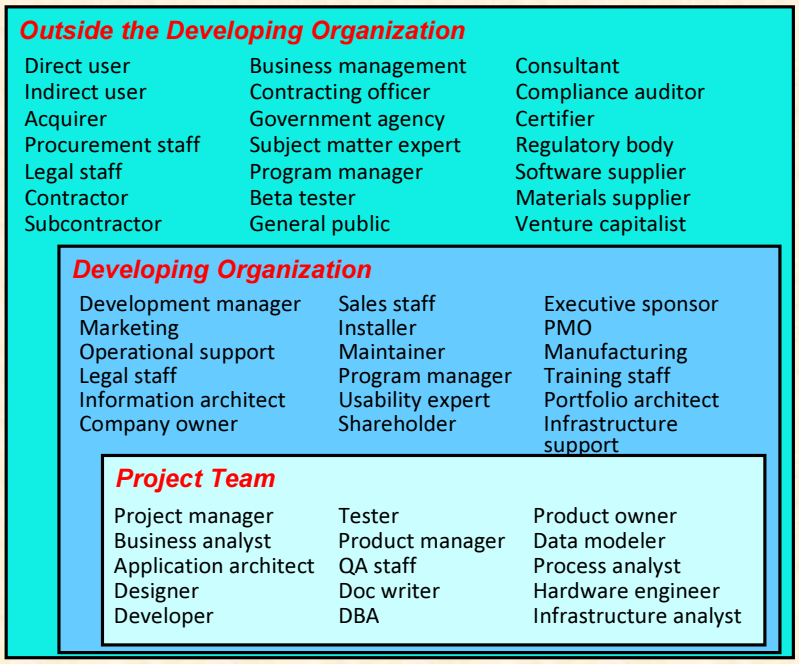
- Understand the need for customer involvement
- Distinguish stakeholders, customers, and users
- Learn customer rights and responsibilities regarding requirements
- Understand what "sign-off" should mean

➢ Chapter 2 of *Software Requirements, 3rd Edition*



- Let's define stakeholders, customers, and users. Although sometimes used interchangeably, these terms refer to different communities of people.
- A stakeholder is anyone who is affected by a project or who can influence the outcome of the project. This encompasses many roles besides customers.
- Customers are a subset of stakeholders (that is, not all stakeholders are customers). Customers derive some benefit from the product, directly or indirectly.
- Users are a subset of customers (for example, someone who makes a purchase decision is a customer but not necessarily a user). Most products have many different user classes, as we'll discuss later.
- Primary users interact directly with the product.
- Secondary users might provide input to the system or get output from it, but they don't work with the product directly themselves.

Some Possible Stakeholders



- Stakeholder analysis is an important part of requirements development.
- Organizationally, stakeholders could be part of the project team, members of the developing organization but not on the development team, or outside the developing organization.
- This slide identifies many of the potential stakeholders in these three categories. Not all of these will apply to every project or situation, of course.
- When searching for potential stakeholders for a particular project, cast a wide net to avoid overlooking some important community. Then you can focus this candidate stakeholder list down to the core set whose input you really need, to make sure you understand all of the project's requirements and constraints so your team can deliver the right solution.

Customer's Requirements Bill of Rights - 1

1. Expect BAs to speak your language.
2. Expect BAs to learn about your business and your objectives.
3. Expect BAs to record information in an appropriate form.
4. Receive explanations of requirements practices and deliverables.
5. Change your requirements.

- The Customer's Bill of Rights and Bill of Responsibilities define the major characteristics of the customer-development partnership that is essential for successful software development.
- You might not agree with all of these, or they might not all apply to your situation. That's fine: adapt them to your needs, or create your own list.
- Then sit down with your customer representatives (could be actual users, or marketing, or someone else) to discuss them and reach a meeting of the minds.

Customer's Requirements Bill of Rights - 2

6. Expect an environment of mutual respect.
7. Hear ideas and alternatives for your requirements and for their solution.
8. Describe characteristics that will make the product easy to use.
9. Hear about ways to adjust requirements to accelerate development through reuse.
10. Receive a system that meets your functional needs and quality expectations.

- Ask yourself:
- Do you think your customers are enjoying all of these rights today?
- Do you think your customers would agree that they are enjoying all of these rights?

In Search of Excellent Requirements

Karl E. Wiegers
Process Impact

"I'll go find out what they want, and the rest of you start coding." This caption from a cartoon is uncomfortably close to the way some software organizations still treat the requirements specification process. Contemporary definitions of "quality" include the concepts of both meeting stated specifications and satisfying the actual customer requirements, which sadly are not always the same thing on a software project.

Perhaps the greatest challenge facing the software developer is achieving a shared vision of the final product with the customer. All stakeholders in a project—developers, end users, software managers, customer managers—must reach a common understanding of what the product will be and do, or someone will be surprised when it is delivered. Surprises in software are almost never good news. Therefore, we need ways to accurately capture, interpret, and represent the voice of the customer when specifying the requirements for a software product [Wiegers, 2003].

As part of a multi-year process improvement effort [Wiegers, 1996], our small software group focused on improved requirements specifications as the starting point for increasing our software quality and productivity. Similar practices have been applied, with considerable success, in many other groups with which I have worked. Our quest for excellent requirements specifications involves six major techniques, which address the challenges of gathering, documenting, and validating user needs:

1. A high degree of customer participation in the development effort through a "product champion" model.
2. Gathering requirements from customer representatives by employing "use cases," which focus on tasks the user must be able to accomplish with the product.
3. Preparation of structured specification documents in a format similar to the IEEE standard for software requirements specifications.
4. Construction of "dialog maps," a variant of state transition diagrams that model complex user interfaces at a high level of abstraction.
5. Extensive use of prototypes to more fully elucidate user needs and explore alternative design solutions for satisfying them.
6. Constructing a requirements traceability matrix to link individual requirements to the design, code, and test elements ultimately constructed to implement and verify them.

While these practices have not been universally applied on every project, we invariably obtain better results when we use them than when we apply our previous, less structured requirements practices. Our experience indicates that these techniques can greatly reduce the expectation gap that often develops between the system the customer really needs to solve his problems, and the system the developer ultimately builds.

Characteristics of Excellent Requirements

High-quality software requirements have several well-defined characteristics, as shown in Table 1. As you write or review requirements specifications, keep these characteristics in mind. Any departures from them are likely to pose problems during the rest of the product development cycle, so correcting these problems early on can save considerable money, time, and aggravation.

Table 1. Characteristics of Excellent Requirements Specifications.

Complete	No requirements or necessary information should be missing. While missing requirements are difficult to identify, sometimes we know we are lacking certain information. Use TBD ("to be determined") as a flag to highlight this gap until the necessary information is obtained.
Consistent	Internal conflicts between requirements must be resolved before development can proceed. You don't know which one (if any) is correct until some further research is done.
Correct	Each requirement statement must accurately state a customer need. Only customer representatives can determine this, which is why it is essential to include customers or their surrogates in reviews of requirements documents.
Feasible	It must be possible to implement each stated requirement within the known constraints and limitations of the system and its environment.
Modifiable	To properly manage requirements, we must be able to maintain a history of changes made in each requirement. This requires that each requirement be uniquely labeled so that it can be referred to unambiguously.
Necessary	Each requirement should document something the customers really need. Avoid the temptation to gold-plate the requirements by adding features the developers are "just sure the users will love."
Prioritized	Assign an implementation priority to each requirement. If the requirements are all equally important, then we lose a degree of negotiation freedom if we have to respond to new requirements added during development, budget cuts, schedule overruns, or loss of project personnel. Consider a three-level priority scheme: High: must be present in the next release Medium: can be deferred to a subsequent release Low: would be nice to have, but we can live without it
Traceable	It is valuable to be able to link each software requirement to its source (a higher-level system requirement, a use case, or a voice of the customer statement). We also want to be able to link each software requirement to the design elements, source code elements, and test cases that are constructed to implement and verify the requirement.
Unambiguous	Ambiguity in requirements is a serious problem. The only way to ferret out ambiguity is to have a group of people representing different perspectives inspect the requirements as a team. Simply passing around the requirements document for comments is unlikely to reveal ambiguity. If a requirements statement is interpreted in different ways by different reviewers, but it makes sense to each of those reviewers, the ambiguity will never surface. (Actually, it will eventually surface, but late in the project, when it can cost a great deal to correct.)
Verifiable	Examine each requirement from the perspective of whether you can devise a small number of tests, or use other verification approaches such as inspection or demonstration, to determine whether the requirement has been properly implemented. In fact, a good guideline for the appropriate level of detail and granularity to write requirements is to make them individually testable.

Worksheets

Worksheet #1: Vision Statement Practice

Write a vision statement for your product, according to the following template.

- **For** (target customer)
- **Who** (statement of the need or opportunity)
- **The** (product name)
- **Is a** (product category)
- **That** (key benefit, compelling reason to buy)
- **Unlike** (primary competitive alternative)
- **Our product** (statement of primary differentiation)