Writing a Requirements Document

For Multimedia and Software Projects

Rachel S. Smith, Senior Interface Designer, CSU Center for Distributed Learning

Introduction

This guide explains what a requirements document is, why it's a good idea to write one, how to write one, and how to use one.

What is a Requirements Document?

A requirements document explains why a product is needed, puts the product in context, and describes what the finished product will be like. A large part of the requirements document is the formal list of requirements.

What Are Requirements?

Requirements include descriptions of system properties, specifications for how the system should work, and constraints placed upon the development process. Generally, requirements are statements of *what* a system should do rather than *how* it should do it. The answers to *how* questions fall into the realm of design. Requirements specifications should not include design solutions (except for interface requirements, which often include embedded design).

Requirements come from end users, from customers, and sometimes from developers. End users tend to state requirements in descriptive or narrative terms ("I'd like a welcome screen with links to the things I use regularly, and I want to be able to select from a couple of different color schemes for the welcome screen") which might need to be broken down into individual requirement statements. Customers, who may well be different from end users, are the people who are paying for the development of the system. Their requirements will often be stated in terms of costs or scheduling issues. Developers might have requirements related to system performance and other technical topics.

It's important to have all these groups contribute to the requirements document to create a fuller description of the system. The practice of including these groups also helps to ensure that everyone is in agreement about what is to be done before development begins.

Requirements documents usually include user, system, and interface requirements; other classes of requirements are included as needed. *User requirements* are written from the point of view of end users, and are generally expressed in narrative form: *The user must be able to change the color scheme of the welcome screen. System requirements* are detailed specifications describing the functions the system needs to do. These are usually more technical in nature: *The system will include four preset color schemes for the welcome screen. Colors must be specified for the page background, the text, visited links, unvisited links, active links, and buttons (base, highlight, and shadow). Interface requirements* specify how the interface (the part of the system that users see and interact with) will look and behave. Interface requirements are often expressed as screen mock-ups; narratives or lists are also used.

What Else Goes Into a Requirements Document?

Besides listing requirements, the document should explain why the product is needed, put the product in context for development, and describe what the finished product will be like.

To put the product in context for development, describe the development tools that will be used, the project budget and staffing situations, and the approximate development schedule. Include details about how the finished product will be distributed or accessed, and any other pertinent information that affects development.

The document should include an overview of the finished product in narrative and/or graphical mockup form for those who want to get a grasp of the project without reading all the requirements. An overview is also useful for people who will be reading the requirements, so that they can have a general understanding of the product before delving into the specifications.

Obviously, the scale of the project will influence the contents and length of a requirements document. Large, complex projects are likely to have more requirements. Smaller, quicker projects will have fewer requirements and will need shorter requirements documents, which can double as project proposals.

Why Write Requirements Documents?

Although writing a complete requirements document is time-consuming, there are many advantages to having one. For one thing, involving major stakeholders in writing requirements helps to ensure that everyone agrees on what is to be done. This can avert misunderstandings down the road and save time that later might be wasted in rework. You can use a requirements document to help set expectations about what will and will not be accomplished in the current development cycle.

A solid requirements document also serves as a guide to development and testing throughout the project. If you have a list of everything your product should do, it's easier to create a product that does everything you wanted (and to test it to see if it really does it all). Your requirements document can also help you schedule time and resources more efficiently and to plan your project more accurately. You'll also have the benefit of knowing when you're done with development.

Here are a few of the problems which can be avoided (or at least lessened) by having a good requirements document:

- building to requirements which do not really reflect the needs of stakeholders
- building a project from inconsistent or incomplete requirements
- making changes to requirements during development, which is costly
- misunderstandings between customers or end users and developers which result in a product that is not what was wanted
- forgetting plans for the project
- feature creep

How Are Requirements Documents Prepared?

A lot of work happens before a requirements document is written. Your project will benefit from the time you spend finding out what the requirements are before writing them down. Once you have gathered and recorded requirements, they should be classified and prioritized. With the list of prioritized requirements and any other project documents you already have, you will be able to compile the requirements document.

Gathering Requirements

Requirements, as stated earlier, should come from end users, customers, and sometimes developers. Most requirements should come from end users. Identify your target user population, then interview some people from that population. It really is important to talk to people from the target population who are not in any way associated with the project; they will think of things you cannot even imagine.

If possible, observe people using something similar to your intended product, and take notes on what works and what doesn't. Make your own observations as well as asking their opinions. Describe your intended product and ask people how they would use it, what features they would need, and what features they would find less useful. You will get some good data this way; just keep in mind that users are generally not developers, and they are not good at describing features they would want before they actually sit down to use a product. However, you can correct for this later during requirements validation.

Interview your customers (the people who are paying for the project) to find out what their priorities are. These will sometimes conflict with user and developer priorities. Record them all and negotiate during the prioritizing phase of requirements gathering.

Other methods for gathering requirements include:

- writing personas and scenarios
- examining project-related documents like proposals, communications, development agreements, etc.
- interviewing developers and designers
- researching the market to find similar products and noting what works/doesn't work about these
- doing task analysis, if the product you are developing facilitates a current tasks users are doing

Recording Requirements

As you gather requirements, use a form to capture them. This will help your requirements remain consistent in form and in terms of the information you capture for each. Basic information to collect:

- a short title
- a detailed description of the requirement
- the person recording the requirement
- the source of the requirement (a scenario, a user interview, the project proposal, etc.)
- a rationale: why is this requirement necessary?
- a list of the stakeholders most affected by this requirement
- initial desired priority, according to the source of the requirement

You may wish to record additional information depending on your project's needs. If your development team is distributed, consider using a web-based requirements tracking system which allows you to record, classify, and prioritize requirements, which are stored in a database.

Classifying Requirements

Once requirements are recorded, the development team should spend some time classifying them. For each requirement, identify the following attributes:

- project component (system, user interface, database, security…)
- type of requirement (system, process, out of scope)
- system components, if applicable (for more complex projects)
- volatility and clarity of requirement
- requirement, assumption, or issue

Project component refers to the part of the product that the requirement affects. For example, a website that dynamically draws information from a database to populate web page templates would have components such as templates, database, user interface, and business logic (or code). Some requirements are applicable to more than one project component.

Type of requirement identifies whether the recorded requirement is something that you need to deal with or not: *system requirements* have to do with how the system is built or functions, so you'll want to deal with those. *Process requirements* have to do with how users use the system; you can't control these. For example, *the user must connect headphones to his computer* is a process requirement for most multimedia projects. You can't ensure that users always do this, so don't include it in your list of system requirements. *Out-of-scope requirements* are those that are impossible or unfeasible to implement during the current development cycle.

Don't throw away process or out-of-scope requirements. Process requirements can inform your development; can you build your product so that it doesn't matter whether the user has headphones or not? Out-of-scope requirements are useful for future project proposals for later versions. Keep both in a separate list so you know where you've been.

System components are applicable for projects that have several different parts. For example, if your project includes an authoring tool that allows nontechnical users to enter information which is then bundled and published as, say, a web page, your system components would include at least *authoring tool*, *templates*, and *publishing logic*. These can be combined with project components for some products.

Volatility refers to how likely a requirement is to change; knowing which are stable and which are less so is very useful. *Clarity* refers to how clearly the requirement is written. Those that are unclear should be refined before the document is completed.

Finally, identify whether what you have recorded is really a requirement, or if it is an assumption or an issue. *Assumptions* are facts which are simply true; if your project is web-based, you can assume that your product will be housed on a web server of some kind. You should still state this in your document. Sometimes the line between assumptions and requirements is hazy. *Issues* are open-ended questions that should be resolved at some point. They may result in requirements or assumptions, or they may be resolved some other way. Flag them as issues to avoid overlooking them.

Once your requirements have been classified, you can arrange them in groups of related requirements. This makes it easier to prioritize them, and gives you a better understanding of which parts of your project will be the most complicated. Typical groupings are by system or project component, with process requirements, out-of-scope requirements, assumptions, and issues pulled out into separate lists.

Prioritizing Requirements

Although your requirements were prioritized when they were recorded, you will likely find that you have more high-priority requirements than development resources. There are probably also requirements in your list which simply should not be implemented, but which seemed desirable when they were proposed.

A second pass at prioritizing will help you select which requirements will be implemented now and which need to be removed to the next version of your project. Prioritization helps to ensure the finished product is as close as possible to what everyone wanted in the first place.

Ideally, prioritization is conducted with representatives of all major stakeholders present, including developers and project managers, end users, and customers. The process of prioritization is a negotiation where all parties discuss their needs and make compromises so that a robust, useful product can be developed on time and on budget. Each requirement is considered and several factors are balanced to assign a final priority:

- desired priority, according to source
- which stakeholders are affected
- how costly the requirement is to implement
- whether the requirement is technologically feasible
- whether the requirement is clearly understood

In reality, the second round of prioritization often happens when developers look at a list of requirements and start identifying those which are especially high-cost or high-risk. It is wise to involve stakeholders other than developers at this stage, though. You may find that a certain high-cost requirement is so important that stakeholders are willing to give up several smaller requirements in order to have it implemented. As you can see, prioritization has an impact on project scheduling.

Once the requirements have been prioritized, you are ready to assemble the requirements document. Another round of prioritization may occur once the first draft is written, so be prepared to go through this process again.

Writing the Document

Armed with your list of prioritized, classified requirements, along with any other pertinent project-related documents, you are ready to write the requirements document. The good news is that at this point, most of it is written already.

The introduction. The document should start with an introduction which explains the purpose of the requirements document and how to use it. State any technical background that is needed to understand the requirements and point less-technical readers to overview sections which will be easier to understand. Explain the scope of the product to define boundaries and set expectations. Include the business case for the product (why is it a good idea?). Finally, include a summary or a list of the most important requirements, to give a general understanding of the requirements and focus attention on the most critical ones.

General description. Devote the next section of the document to a description of the product in nontechnical terms. This is valuable for readers who want to know what the project is about without wading through the technical requirements. Include a narrative which gives readers an idea of what the product will do. Identify the product perspective: the need(s) the product will serve, the primary stakeholders, and the developers. Describe the main functions of the product (what activities will users be able to do with it?) and describe typical user characteristics. Explain any constraints you will face during development, and list your assumptions and any technical dependencies.

Specific requirements. Include your requirements list here, divided into whatever groupings are convenient for you.

Appendices. Include, if you wish, any source documents such as personas and scenarios, transcripts of user interviews, and so on.

Glossary. Explain any unusual terms, acronyms, or abbreviations (either technical or domain-specific).

References. List references and source documents, if any.

Index. If your document is long, consider including an index.

Document Template

There is a requirements document template available for your use:

http://www.cdl.edu/resources/writing-requirements.html

You may freely copy, alter, use and distribute this template. Just don't sell it.

How Is a Requirements Document Used?

By this point in the process, you should have clarified what your product will actually be like. If you were careful to include them in the process of gathering requirements, you should have buy-in from all your major stakeholder groups. Perhaps some of your findings were surprising; hopefully you were able to incorporate the surprises into your project plan.

Once the first draft of your requirements document is complete, don't file it and forget it. Part of the point of writing one is the process of unearthing, recording, organizing, and prioritizing requirements, but the document is useful during development as well. First, however, it should be validated, and probably revised.

Validating Requirements
Go back to end-users and customers with the draft requirements document and have them read it (mostly the narrative sections). Ask if this is what they had in mind, or if they think any changes are in order. It's much easier to make changes now, while your project is still words on paper, than after development has begun. Take their comments and make changes; you may need to revisit the previous stages (recording, organizing, and prioritizing), but it will be easier this time. The vital thing is to come up with a requirements specification that meets the needs of the major stakeholders. Repeat the process until there is agreement, and get representatives of each stakeholder group to actually sign off on the document.

During Product Development
During development, refer to the document for guidance. Interface designers, instructional designers, and developers should use it as a map to direct their efforts. If someone's work is called into question, check with the requirements document to decide which direction to proceed. This isn't to say that you have to treat it as inviolate. It's possible that small requirements changes will continue to occur. Just be sure to get stakeholder agreement before altering the specification.

During Product Testing
Continue to check the ongoing development process against the requirements; does the emerging product do what it ought? When the product is in testing, check it against the requirements again. Was anything omitted? If you're having problems with a feature, check with the requirements document to see if there is a way around the problem. Sometimes developers add features which don't need to be there, and in a time or budget crunch, you can use the requirements specification as a reason to put off an extra feature.

After Release
Keep track of the "wish list" and "version two" requirements. You may have the makings of a grant proposal there!

Resources
A website has been prepared to accompany this workshop:

http://www.cdl.edu/resources/writing-requirements.html

It includes requirements document templates, this workshop handout (PDF), the accompanying PowerPoint slide presentation, and links to books on requirements engineering.

Reference
Sommerville, Ian & Pete Sawyer. (1997). Requirements Engineering: A good practice guide. West Sussex, England: John Wiley & Sons.