Writing a Software Requirements Document

Tanya Berezin

Table of Contents

Should You Read This Paper?

This paper discusses the purpose and contents of a requirements document for a business application. It is an introduction to the subject and will be most helpful to you if any of the following applies to you:

- you are responsible for collecting requirements for a business application
- you are leading a business application development project
- you are not sure what a requirements document ought to look like or even if you need one
- you are not sure what to do with a requirements document even if one miraculously appeared on your desk tomorrow

This paper will help you write a professional requirements document. Once you feel you understand what a requirements document is, I urge you to start reading more advanced material; some of the books are listed at the end of this paper.

If you are an experienced requirements analyst and/or project manager you may want to skim this paper to see if you can add any of the information here to your own "bag or tricks". You also may want to review the bibliography at the end of the paper to see if you have missed any of the great books listed there.

What Is a Requirements Document?

The software requirements document is a written statement of what the software will do. This seems quite a dull statement but it is worth examining a bit closer.

➔ **Requirements document states *what* the software will do. It does not state *how* the software will do it.**

What the software does is directly perceived by its users – either human users or other software systems. When a user performs some action, the software responds in a particular way; when an external system submits a request of a certain form, it gets a particular response. Therefore you and the users must agree on actions they can perform and response they should expect. This common understanding is captured in the requirements document.

How the software responds to the agreed upon request is *not* addressed in the requirements document. For example, the requirements document does not include screen layouts, database schemas, descriptions of communication layers – in short, no statements of design of any sort. For example, it is a requirement for a word processing application to be able to open an existing file. It is a design issue whether to build a customized file selection tool or use a platform-standard file selection tool.

This is not to say you won't seek users' input on some of the design, most especially on user interface design, but it is very important to recognize and

respect the boundary between the statement of requirements and how requirements are implemented. Design is the responsibility of the development team they should be free to choose the most appropriate way to satisfy all aspects of the requirements – features, performance, usability, etc. Usually the most appropriate way is the most simple way but sometimes other considerations may affect design decisions – such as opportunities for design or code reuse, for example.

➔ **Requirements document is a *written* statement.**

The requirements document is not a collection of notes and Post-It's, e-mail discussions, or accumulated knowledge in someone's head. They all are useful as supporting information but they cannot be distributed easily for review or verified.

You can give a written statement to other people to read and discuss. The document can serve as the basis for an agreement between you and your customers and it can be used by the developers and testers to implement the application and verify that it meets the original agreement. There are many great things you can do with a written statement.

Why Bother with a Requirements Document?

The answer is (or ought to be) pretty obvious: if you haven't written down what the application is supposed to do, how will you know what to develop and how will you manage your customers' expectations?

Obvious though the need may seem, you often will have to preach the gospel of requirements documentation far and wide in your organization unless your company already follows mature development practices. Even if your IS organization does not need convincing, your customers (either internal or external) may wonder if you are wasting their time and money writing a requirements document when you really should be coding!

➔ **The main purpose of a requirements document is to serve as an agreement between the developers and the customers on what the application will do.**

In many cases this agreement is enforced via a legally binding contract. Even if this is not the case for your application, I strongly recommend you view the requirements document as a contract between you and your customers. Moreover, you should strive to instill the same attitude among your customers. If everyone treats the requirements document as a software development contract, all parties are more likely to have common expectations for the application – a very necessary thing for your project to succeed.

The requirements document brings the following additional benefits:

- the customers can see early on if their needs will be met
- the developers can estimate the effort involved in creating the application
- the development project leader has a basis for a project plan
- the quality assurance people have a basis for testing the application

Do I Have to Write a Requirements Document?

In case the previous section did not quite convince you that you need a requirements document for your next application, this section offers a simple test that will tell you if you must write a requirements document or you can "wing it".

It is true that in some cases you can get away with not having a formal requirements document: if your application is small enough to be developed by a single person and the user population is also very small (a few people or a single workgroup). When this is the case, it is possible to achieve a common understanding of what the application will do without committing the knowledge to paper. In all other cases you really need a written requirements document.

➔ **You must have a requirements document if any of the following is true for your application:**

- **it will take more than about a calendar month from conceiving the project to putting in into production**
- **more than one person will work on the application**
- **more than one workgroup will use it**

Who Uses the Requirements Document and Why?

There are many distinct roles in each software development project. While the same person often plays more than one role on a given project, it is most useful to consider how every role on the project uses the requirements document.

Below I list the project roles and the reasons for each role to use the requirements document. Remember that each of these roles uses the document in a different way; indeed some may use only some parts of the document. When we discuss the structure of the requirements document in later sections I will point out which roles primarily use each section of the document.

Role in a Software Development Project	Reason for Using the Requirements Document
Development Project Leader	♦ Scope the project; divide the project in phases ♦ Obtain agreements from the business expert, project sponsor, and the development manager on the scope and schedule for phases ♦ Track development progress
Requirements Analyst	♦ Elicit and document business requirements ♦ Test the application against the requirements

Development Team Member	♦ Design and code the application
QA Specialist	♦ Verify that application conforms to requirements
User Documentation Specialist	♦ Produce user documentation
Legacy Support Specialist	♦ Support legacy integration needs of the project
Maintenance Team Member	♦ Learn the application to take over the production support from the development team ♦ Support users in production
Development Manager (i.e., the person to whom the development project leader reports)	♦ Understand planned phases and coordinate with the development of other applications ♦ Track development progress
Project Sponsor	♦ Provide the motivation for the project ♦ Sign off on the planned phases and their scope
Business Expert (often designated by the project sponsor)	♦ Confirm that the requirements reflect the business needs ♦ Sign off on the planned phases and their scope

General Principles in Writing a Requirements Document

The previous three sections have given you the motivation to produce a requirements document: you know what purpose it serves, what benefits it offers, when you absolutely must have one. You found some users for it so you know it won't gather dust on a shelf.

Next we will consider what a requirements document looks like but before we dive into details I will introduce some general ideas that you should have in mind while working on a requirements document.

Avoid Unnecessary Work

First I'd like to state an idea I call the Least Work Principle:

➔ Do not do any work that costs you more than it is worth to you.

This principle is very general and it requires conscious interpretation in every work situation to be useful. For example, I always clear my desk before I leave office for the day because seeing a desk piled high with papers first thing in the morning ruins my mood for the rest of the day. For me the Least Work Principle in this situation dictates to put stuff away at the end of a day. Most people tolerate a messy desk just fine (in other words, a tidy desk is worth nothing to them) so quite correctly they do not spend any time cleaning it. They may not

even know it but they apply the Least Work Principle when they leave their desk messy.

This is how the Least Work Principle applies to writing requirements documents.

➔ **Tailor the structure and the level of detail in your requirements document to the nature of the application.**

You need a detailed document for:

- large application (many interfaces, screens, lots of features)
- application with many users or many user workgroups
- application that is critical to the business (e.g., online banking, or a reservation system for an airline)

If none of the above is true for your application you need a brief requirements document.

By consciously adhering to this principle you will be preserving not only your own sanity but also the sanity of everyone who uses the document.

Use Iterations

Iterative approach to writing software is widely accepted but it applies to writing the requirements document just as well. Here is how it works:

- In the beginning of the project, set out the structure of the requirements document. At this point the document will have little more than section headings, but that's useful since it gives structure to your work.
- Start filling in the objectives section right away; discussion below shows why this should be your first step
- As you talk to users and learn more requirements, capture them in the appropriate sections; make sure that the business process and user role sections get filled in early on
- Keep going!

Since you approach your work in iterations, you always have a version of the entire requirements document. You can talk about your project to anyone at any time and be able to disseminate information and collect information a lot more efficiently. Also, you always know where the holes are and what to do next.

Verify Information

When you are collecting requirements it is very important to verify all of the facts by interviewing customers with differing points of view – users and management; users in different parts of the organization; etc. This does not mean you should not believe your sources. But you always need to remember that different people in the same position and especially people in different positions have a different view of the business needs and business practices as far as your application is concerned.

If your users are in divisions that use different business practices (as is often the case for companies with offices in different countries), you must get the information from all divisions that will use your application.

Write to Read

We already said that the main purpose of the requirements document is to serve as a contract between you and your customer about what is to be done. This means you are writing the requirements document to be read and understood by others, not just yourself. You are responsible for making it easy to read and absorb the document.

➔ Write simply.

I do not presume to teach you how to write technical documents (I am as likely as the next "techie" to write dense documents filled with jargon). I simply list below some of the "tricks of the trade" that generally help in making the document more readable.

- Use short sentences, short paragraphs, bulleted lists – anything that will let your readers draw a breath and help them focus.
- Write in the active voice whenever possible. "The system will pass the data set to the telephony application" is much more clear than "the data will be passed to the telephony application".
- Make sure the sections of your document follow each other in a logical order.
- Illustrate your points with charts and tables. I am not advocating pretty graphics for their own sake, but some things are much easier to convey with a picture. Some examples are:
  - ◊ charts to show the hierarchy of user roles
  - ◊ charts to represent workflows
  - ◊ block diagrams to represent external interfaces of your system
- Avoid computer jargon; define the terms you do need to use.
- Include the table of contents and index. The former makes the structure of your document clear and so carries information valuable to the reader. The latter makes it easier to find things. If you have many figures and tables, include a separate table of contents for these.
- Use the spellchecker!

➔ Partition the requirements document into several when necessary.

For large and / or complex applications it is often convenient to split the requirements document into two or more documents. For example, if your application interacts with a legacy system, you may want to put the legacy integration requirements into a separate document because they tend to be lengthy, very technical, and are weakly related to the rest of the requirements. If your application requires elaborate security and / or auditing features you may want to pull those requirements into a separate document. User Interface, and external application interfaces are examples of potential separate requirements documents.

Sections of a Requirements Document

This part of the paper describes the purpose of each section of a requirements document and provides some advice on the usage of each section. You may want to expand the list of sections below for your project but it should suffice for the development of most business applications.

Each requirements document consists of at least two parts – an overview and a description of the system's functionality. Often the document includes appendices for material that needs to contain more details than the rest of the document or for material that does not fit anywhere else but still needs to be included. Below I address each of these parts in turn.

In a separate section I include a table that lists which project roles use which section of the requirements document.

Part I – Application Overview

This part of the requirements document serves to present the "big picture" of the application. Here you lay out the objectives of the application, how it fits into the business process of the company, and how it relates to other software systems. The sections listed below should be included in this part of the requirements document.

Objectives

In this section you state the commonly accepted objectives of the project.

You must determine the business objectives of the project early on; without clear objectives your project has little chance to succeed anyway so it does not make sense to move on until the objectives are agreed upon.

➔ First answer the question: Why are we doing this?

To elicit the objectives, ask the business expert, the development manager, and the project sponsor the following questions:

- What business objectives of the company will this project help achieve? Possible objectives might be reducing costs, improving the customer service, simplifying the work flow, replacing obsolete technology, piloting a new technology, and many others. Also, make sure you understand exactly how the proposed project will help accomplish the stated objective.
- Why are we doing this project now? What will happen if we do it later? What if we do not do it at all?
- Who will benefit from this project? Do the people who will benefit from it consider it the most important improvement that can possibly be made at this time? Should we be doing a different project instead?

Write down everybody's answers and make sure they do not contradict each other. If they do, keep working on defining the objectives in such a way that all stakeholders agree on them.

It is your responsibility to make sure everyone understands the objectives and remembers them. Use every opportunity to keep this information in front of all parties – include it in any communication you have with anyone about the application.

Most importantly, you yourself must be aware of the objectives at all times and you must be sure all major requirements are directly related to meeting one or more of the objectives. If you find yourself under pressure to accept requirements not related to the project's objectives, you must re-verify the objectives with the project sponsor, the business expert, and the development manager.

Business Process

In this section you describe the business process and how your application will be used in this context.

In some cases you will need two sections of this sort – one describing the existing business process using existing systems and the other describing the future business process using the system you are developing. This happens any time the business process will change once your system is introduced. When this is the case, the purpose of describing the existing business process is to have a basis of reference to explain the new business process.

If the business process won't change when your application is introduced you should be able to describe it in a single section. However, in this case be sure you understand and communicate to others what value your application brings to the customers. This question should be answered in the Objectives section.

User Roles and Responsibilities

In this section you describe who the users are and how the system fits into what they do.

You need to list all users for your system in terms of user roles. *Typically each individual performs multiple roles in the course of his work since his job involves meeting multiple business objectives. A user role is related to meeting a specific business objective. When gathering requirements it is most useful to consider roles since you will want to focus only on those business objectives that are relevant to your application.*

For each role you need to list the tasks that involve the use of your system (directly or indirectly). You also need to describe the relationships among the tasks for each individual user role and the hand-offs from one role to another. This is usually represented as a workflow diagram.

Consider time in describing tasks and their relationships – different sets of tasks may be performed at different times (daily, monthly, etc.) and several workflow diagrams may be needed.

Once you have written the Objectives, Business Process, and the User Roles and Responsibilities sections, give them to the business expert to read. If you and he agree on what's written, congratulations! You are well on your way to understanding what actually needs to be done.

Interactions with Other Systems

In this section you describe how your application relates to other existing systems.

If your application is a sub-system of a larger system, this section should have two sub-sections. In one you describe the interactions between your sub-system and other sub-systems within the larger system. In the other you describe the interactions between your application and other systems outside of that larger system. These other systems may include legacy and non-legacy systems.

At the very least you should include a diagram showing the interfaces between your system and others. It should show the relationships between systems and what information is passed between them. It is best to add a verbal description of how your application fits in with everything else in terms of application responsibilities.

Replacement of Legacy Systems

If your application replaces any legacy systems, in this section you describe the bootstrapping required for switching production from the legacy to the new system.

This bootstrapping may be quite involved and need a lot of technical detail. You may want to include a very short summary section in the requirements document and write a separate legacy replacement requirements document.

Production Rollout Considerations

In this section you describe the strategy for production rollout.

In addition, either this section, or an appendix in the requirements document, or a separate document should include the discussion of populating the system data for rollout and the discussion of the expected data and transaction volume.

Terminology

In this section you define the business terms used in the requirements document.

You should include this section even if at first it seems like a waste of everyone's time. Once you show it to people you may be surprised to learn that not everyone understood the terms the same way after all!

Part II – Functional Requirements

This part of the requirements document states in a detailed and precise manner what the application will do.

On different projects I have used two ways of expressing the functional requirements: with the help of use cases or as free text. In this paper I will not

focus on the use case approach; at the end of the paper I list some books that describe the use case approach in detail.

This part of the paper lists the sections that need to be included if you choose to capture functional requirements in the form of free text.

Statement of Functionality

In this section you state **precisely** *what the application will do.*

This part, more than anything else in the requirements document, spells out your contract with your customers. The application will include all functions listed here and will not include any of the functions not listed. Since this is your contract, think very carefully about what to include and what to leave out.

In this section you must use as precise language as you can since the developers will use it to code the application. When reviewing this part with other people you should pay extreme attention to removing any possibility for ambiguous interpretation of any of the requirements.

This section will consist of multiple sub-sections, each devoted to one area of functionality. There are several principles you can use to decide how to structure this section.

If your application has several distinct categories of users, you can list the requirements by user category. User categories may be defined in terms of their job title (clerk, manager, administrator), the frequency with which they will use the system (heavy or casual), the purpose for which they will use the system (operational decisions, long-term decisions), etc. As long as each category of users uses the system in its own way, structuring the requirements based on user category will make sense.

If your application deals with several kinds of real world objects, you can list the requirements by object. For example, for a reservation system a booking is an important object and you may want to list all requirements pertaining to bookings in one sub-section.

One of the most common approaches is to list the requirements by feature. For example, features of a word processing application are file management, formatting, editing, etc.

IEEE has developed some templates for structuring a requirements document. You can find more ideas there; I reference the document at the end of this paper.

While most of the functionality addressed in this section will be specific to your application, some areas of functionality are so common to business applications that you should make sure you do not omit them accidentally. I list them below[1].

[1] If you are using use cases to express functional requirements you also need to remember to address the issues listed below.

- **Security**

  In this sub-section you list any requirements related to security, i.e., who is authorized to use any parts of your application.

  Sometimes people confuse security with pertinence: people may tell you that so-and-so should not be able to access certain part of the application when they really mean that so-and-so has no need to see that part of the application but there is no breach of security if he does. Obviously, these two interpretations can make a difference to the design of your application, so be sure you understand what the real security concerns are.

  If the security requirements are very complex you may want to write a separate document to describe them.

- **Auditing**

  In this sub-section you list any requirements for auditing features.

  Auditing usually means being able to tell who made changes, when they were made, and what they were. Be sure you understand precisely what your organization thinks "auditing features" are.

- **Administration/Customization of the Application**

  In this sub-section you list any requirements related to the users' ability to modify the application or to perform any administrative functions (such as creating new user id's or other appropriate functions).

- **Reporting**

  In this sub-section you list the reporting requirements. Alternatively, you may want to write a separate document if the reporting functionality is extensive.

  Reporting requirements generally specify what information is presented in the reports, how frequently the reports are produced, and to whom they are distributed. They do not specify what the reports look like – this is a design issue.

In addition to listing requirements related to the functionality of the application, the functional requirements part of your document includes the following general sections.

Scope

In this section you state what functionality will be delivered in which phase.

You should include this section if your development will consist of multiple phases. As an alternative to this section, you can note the planned project phase for each feature in the functionality statement section. Usually it is better to include a separate scope section for easy reference and communication.

Whether or not you use a separate scope section, you may want to prepare a section or a separate document that will list in detail what is not *in scope for the initial release. This will catch everyone's attention and will bring out any*

misunderstandings about scope when there is time to address them. You may want to write a section like this for all subsequent major releases of your application.

Performance

In this section you describe any specific performance requirements.

You should be very specific and use numeric measures of performance. Stating that the application should open files quickly is not a performance requirement since it is ambiguous and cannot be verified. Stating that opening a file should take less than 3 seconds for 90% of the files and less than 10 seconds for every file is a requirement.

Instead of providing a special section on performance requirements, you may include the relevant information for each feature in the statement of functionality.

Usability

In this section you describe any specific usability requirements.

You need to include this section only if there are any "overarching" usability goals and considerations. For example, speed of navigation of the UI may be such a goal. As in the previous section, use numeric measures of usability whenever possible.

Concurrency

In this section you describe any specific concurrency requirements.

Instead of having a separate section, you may choose to include the concurrency requirement for each feature in the statement of functionality.

Part III – Appendices

Appendices are used to capture any information that does not fit naturally anywhere else in the requirements document yet is important. Here are some examples of appendices.

Supporting and background information may be appropriate to include as an appendix – things like results of user surveys, examples of problems to be solved by the applications, etc. Some of the supporting information may be graphical – remember all those charts you drew trying to explain your document to others?

In some cases you find yourself explaining some part or another in more technical detail than the surrounding material. This part can be separated out into an appendix.

Appendices can be used to address specialized audience. For example, some information in the requirements document may be more important to the developers than to the users. Sometimes this information can be put into an appendix.

Who Needs What? Summary of Purpose and Usage of the Sections of the Requirements Document

This summary lists all sections of the requirements document described so far. For each section the table contains the main purpose of the section, whether you must include this section in the document, which project roles will use it, and whether you should include it in a summary document whenever you produce one.

Obviously, the requirements analyst and the development project leader use all sections of the document. For the sake of brevity I omitted these two roles from the lists below.

Part I – Overview

Section	Purpose	Required?	Project Roles Who Use It	Include in Summary?
Objectives	State the commonly accepted objectives of the project.	Yes!	All roles	Yes!
Business Process	Describe the business process and how your application will be used in this context.	Yes	All roles	Yes
User Roles and Responsibilities	Describe who the users are and how the system fits into what they do.	Yes	development team members, business expert, QA specialist, user documentation specialist, maintenance team members	Yes
Interactions with Other Systems	Describe how your application relates to other existing systems.	Yes	development team members, development manager, project sponsor, business expert, QA specialist, user documentation specialist, maintenance team members	Yes
Replacement of Legacy Systems	Describe the bootstrapping required for production.	Yes	development team members, development manager (summary only), project sponsor (summary only), business expert, QA specialist, legacy support specialist, maintenance team members	Yes
Production Rollout Considerations	Describe the strategy for production rollout.	Yes	development team members, development manager, project sponsor, business expert, QA specialist, legacy support specialist, maintenance team members	Yes

Terminology	Define the business terms used in the requirements document	Yes	All users	Yes[2]

Part II – Functional Requirements

Section	Purpose	Required?	Project Roles Who Use It	Include in Summary?
Statement of Functionality	State *precisely* what the application will do.	Yes	development team members, business expert, QA specialist, user documentation specialist, maintenance team members	No
Scope	State what functionality will be delivered in which phase	No	development team members, development manager, business expert, QA specialist, user documentation specialist, maintenance team members	Maybe
Performance	Describe any specific performance requirements	No	development team members, business expert, QA specialist, user documentation specialist, maintenance team members	No
Usability	Describe any specific usability requirements	No	development team members, business expert, QA specialist, user documentation specialist, maintenance team members	No
Concurrency	Describe any specific concurrency requirements	No	development team members, business expert, QA specialist, user documentation specialist, maintenance team members	No

How to Get Others to Read the Requirements Document?

So you've gone to all the trouble and wrote a requirements document. No doubt you feel very virtuous; moreover, you know what needs to be done. Now comes the hard part – you need to get your customers to read it so you and they have a common understanding of what the application will and will not do.

➔ **The best requirements document is worth nothing if it is not read by all appropriate parties.**

Failure to have a common understanding of the requirements can doom your project and most certainly will make it late. Have you ever heard the customer say during the system acceptance test: "But I thought it was going to do X, Y, and Z!"?

[2] You may want to omit this section from the most terse, "executive level" summaries.

Of course, while you were writing the document, you discussed parts of it with others and showed them some excerpts. You were planning on showing them excerpts and drafts, weren't you? Yet there comes a time when a lot of people must bite the bullet and read a document which by now is a couple dozen pages long. A section earlier in this paper summarized which groups of people need to read each section.

Here are some of the techniques that may help you to get people to read what they must:

- Use written summaries – pull out sections of the master document into a made-to-order document. It takes only a few minutes but it lets you target the information to your audience. Also, the result is a much thinner paper which people are more likely to read.
- Use stand-up presentations – the chunks of information will be smaller and easier to digest and you can tailor the contents to the audience. Also, at least for the duration of the presentation you will have your audience's attention.
- Use document walkthroughs or more formal document inspection mechanisms (this works especially well with the development team). At its simplest, you basically read the document together and focus on how the readers understand what you wrote.
- Focus on the business expert – he/she really needs to read and digest it all. Give it to her in small chunks, cajole, read it with her, hand out candy, whatever – just make sure she reads it. Here is a hint: if you get no criticisms or questions from the business expert, most likely she did not read the document or was paying no attention when she read it.
- With management you need to make sure they digest the following things. First, elicit their comments on how the project fulfills the business objectives and how it fits into overall business process – their perspective might be different from the one you got from the lower-level people. Second, make sure they understand the scope of the phases so their expectations for each deliverable are appropriate.
- Some people use system prototypes as a way to present requirements. This may work in some cases but I prefer to use prototypes as a mechanism to elicit requirements earlier in the process.

You will need to wage this information dissemination campaign several times during the project. The most notable such milestones are:

- Requirements sign-off
- Presentation of a quite fully featured on-screen prototype
- System testing
- Deployment

Reflecting Changes in Requirements

In a properly run project there is a time when the requirements are signed off by the development manager and the customer. But let that not fool you: requirements will continue to change after this sign-off, most likely, right up to the system acceptance testing or even up to the system deployment. Yet the

requirements document still is the contract between you and the customer on what the system will do, so it must reflect the changes.

➔ **You must continue to update the document as the requirements evolve.**

As you continue to update the document (either before or after the sign off), you need to keep track of how it changes over time. Often the changes in requirements reflect changes in business needs or in your understanding of business needs. Recording them will help you understand the business and the customers better. Recording the changes also documents them for everyone to see so you can always point to the document when someone begins to wonder why his favorite feature is not there.

You should use your discretion in deciding how detailed the tracking should be (remember the Least Work Principle?). At a minimum, you should keep an archive of the versions of the document that were distributed to a wide audience, for example, at the milestones listed in the previous section.

In addition you should maintain a revision history at least for major requirements. If the requirements change dramatically between the conception of the project and the requirements sign off, you should consider including a separate section that describes the major changes and reasons for them.

Documenting Requests for Enhancements

There does come a time when the requirements for the initial release of your application are frozen. Usually it happens after the system acceptance test which is the last chance for the users to lobby for some changes to be introduced in the upcoming release.

At this time you need to begin maintaining the list of requested enhancements. Below is a template for tracking requests for enhancements.

Date	Enhancement	Requested by	Notes	Priority	Release No/ Status

If the enhancements are very substantial, you may consider writing a full-fledged requirements document for the new release.

In addition to documenting major enhancements I strongly recommend using an off-the-shelf defect tracking system for cataloging bugs and minor enhancements as soon as your application begins the QA process. In some organizations this will be the responsibility of the QA group, in others the project leader or the requirements analyst will perform this task.

Tracing Requirements

Tracing requirements means relating specific requirements to other project documentation, especially to the following:

- a requirement to its source (backward tracing)
- one requirement to another (traceability matrix)
- a requirement to design document, code, user documentation, or project plan (forward tracing)

Backward tracing allows you to document the source for each requirement – for example, you can note that a specific feature was requested by a certain user or a group of users. This information can help you judge the priority of the requirement which can be very important in development project planning.

Creating a traceability matrix can help you evaluate the effect of changes to requirements. If someone requests a change in requirement A and you know that requirement A is related to requirement B which in turn is related to requirements C, D, and E, then you know the change may be costly and you will examine it carefully before accepting it. Once you accept it, you will be thorough in making all changes implied by the requested change.

Forward tracing is invaluable in planning and implementing the development project and in making sure all changes to requirements are properly reflected in the design, code, project plan, etc. For example, tracing requirements forward to design will ensure that you have solutions for all requested features. For another example, tracing requirements to the project plan will allow you to always know how much work has been completed and how much remains.

Obviously, for tracing requirements forward and to each other you must find a way to enumerate all specific requirements and to label them in some way.

Conclusion and Further Reading

I hope this paper gave you some ideas about structuring the next requirements document you will work on. But this is just a very brief introduction to the subject of requirements engineering. Below I list some of my favorite books to start off your reading. In addition, I strongly recommend books by Gerald Weinberg who does a lot of work in the area of systems analysis.

General Systems Thinking

Gerald Weinberg *Rethinking Systems Analysis and Design*, Dorset House Publishing, 1988

Gerald Weinberg, Donald Gause *Are Your Lights On? How to Figure Out What the Problem Really Is*, Dorset House Publishing, 1990

Eliciting and Analyzing Requirements

Gerald Weinberg, Donald Gause *Exploring Requirements: Quality Before Design*, Dorset House Publishing, 1989

Naomi Karten *Managing Expectations: Working with People Who Want More, Better, Faster, Sooner, NOW!*, Dorset House Publishing, 1994

Guides to Good Practice

Karl E. Wiegers *Creating a Software Engineering Culture,* Dorset House Publishing, 1996

Ian Sommerville, Pete Sawyer *Requirements Engineering: A Good Practice Guide,* Wiley & Sons, 1997

Use Case Methodology

Ivar Jacobson et al *The Object Advantage: Business Process Reengineering with Object Technology*, Addison Wesley, 1994

Ivar Jacobson et al *Object-Oriented Software Engineering: A Use Case Driven Approach*, Addison Wesley, 1992

Alistair Cockburn *Structuring Use Cases with Goals*, JOOP/ROAD Sep and Nov 1997[3]

Standards

IEEE developed a standard for requirements specification. It includes some helpful templates for structuring the document.

IEEE Std 830-1993 *Recommended Practice for Software Requirements Specification,* IEEE Standards Collection: Software Engineering. The Institute of Electrical and Electronics Engineers, 1994.

Software Tools

There are several tools to assist with requirements engineering. While I do not have enough personal experience with them to recommend any, here are two examples:

- RequisitePro™ from Rational Software Corporation
- Caliber-RM™ from Technology Builders, Inc.

Author Biography

[Here is where you tell them about all your experience that had made you an expert on this topic. This is where you let people know that you know that working on a software development project without a spec is hell on earth.]

[3] The paper and a companion template are available on the Web (http://members.aol.com/acockburn)