

Software Requirement Definition: How to Avoid Disasters in Software (part 2)

Introduction

The software world has changed a great deal since 2002, but one thing remains the same: too many projects fail unnecessarily. This year, like last year and the year before it, billions of dollars will be lost to poor planning. In fact, losses will likely be greater this year than ever before.

Now – more than ever – an understanding of software requirements definition may be the difference between a software success and just another software statistic.

Recap

In Part 1 of this series, we discussed the consequences of failing to define software requirements. According to the Standish Group, 53% of software projects finish late, over budget, or lacking functions, and 31% are abandoned. For 2 out of 3 of these projects, requirement errors are to blame.

We also discussed some of the reasons for these errors which included:

- Difficulty in thinking in the abstract
- Inability to predict the future
- Techs diving straight into the code
- Deadline pressure
- Technology in search of an application
- Estimation problems
- Lack of user focus and dialogue
- No need for analysis; we're buying a package

Having isolated the problem and the consequences, we began presenting a solution. The first step in the requirements definition process must always be identifying your need. Do you have a problem that needs solving with software? If you do, what is it? Falling sales? Escalating costs? Low profit? Are you trying to increase market share? These types of questions need to be asked and answered. Just as importantly, you need to measure the importance or severity of the need and find out what the underlying causes are. Only then can you decide if – and how – your need can be met.

Meeting the Need

The challenge is now to weigh up the alternative solutions and get business buy-in.

Identifying and Assessing the Alternatives

For every need, the software world seems to offer a thousand solutions. Weeding through the alternatives is a project in itself, so you need to make sure you approach it in a structured manner.

You need to identify the constraints and objectives of all serious alternatives. Constraints are the things that stand between you and implementation of that particular solution, including budget, technology, environment, people, and skills. Objectives are the things that you expect/want the solution to give you. Make sure you give this careful consideration. What are the basic criteria? Can we measure our success? Are we trying to improve gain or reduce pain?

This analysis will allow you to roughly outline the obvious risks and scope of each solution (including broad costings and timeframes). Then you've got the resources you need to make an informed comparison of the alternatives, and decide which you'd like to recommend.

Recommending a Solution

In terms of getting business buy-in this is probably the most important step of all. Think of it as your chance to present your vision for the project and to define the scope of the solution. But always – *always* – make sure you present your recommendation as a business case. If it doesn't make sense to the business, it's just an expense!

Your recommendation is in the form of a Vision and Scope document, and should include:

- Concise statement of business requirements
- Project description and goal
- High level scope (major features)
- High level cost/benefit analysis
- High level risks

Tips for Selling to Management

1. Engage management from step 1
2. Manage perceptions and expectations
3. Listen and respond to objections
4. Keep the focus on business, not IT

- High level assumptions
- Exclusions

If and when you get business buy-in, then the real fun starts...

Defining the Requirement

Once you've decided how your business need should be filled, and you've got management support, you need to take your first steps toward that goal.

Although your work to this point may have been painstaking, you've really only been dealing at a high-level. Now it's time to get down to nuts and bolts.

What are the intimate requirements of your solution? Exactly what does it need to do?

This is a big question, and it's not something you can answer by yourself. In most projects, a great many people are involved in gathering requirements:

- | | |
|-------------------|------------------------|
| • Analyst | • Customer |
| • Project sponsor | • Facilitator |
| • Scribe | • Project manager |
| • IT specialists | • Visiting specialists |
| • Users | • Business people |

Think carefully about who will fill these roles. In particular, think carefully about the Analyst and Customer roles. They're the ones who'll have the greatest involvement.

Your Analyst must:

- Speak the customers' language
- Learn about the customers business and objectives
- Prepare a Software Requirements Specification (SRS) which accurately maps the customers' requirements
- Accept feedback from customers on the SRS

Your Customer must:

- Educate analysts about the business
- Allocate sufficient time to provide and review requirements
- Set priorities
- Make business decisions

Remember, your 'customer' isn't necessarily a real-world customer. They may have been at one time, but normally they're a customer advocate – someone who knows what real-world customers need to do every day.

Some Techniques for Gathering Requirements

There are a lot of methods around for gathering requirements. You've no doubt used some – or all – of them yourself at some stage. Here's a list of some of the most effective.

Technique	Basic Format	Typical Participants	Benefits	Problems	Comment
Workshops	Key stakeholders are gathered together for a short intensive period typically 1-2 days. The requirements are extracted by a facilitator and instantly minuted by a scribe.	<ul style="list-style-type: none"> • Project sponsor • Facilitator • Scribe • Project manager • IT specialists • Visiting specialists • Users • Business people 	<ul style="list-style-type: none"> • Team building / Encourage buy-in • All voices heard • Expose political issues early • Instant feedback 	<ul style="list-style-type: none"> • Timekeeping – start, back from breaks, relevant topics • Grandstanding • Non-contributors / Shy / Politics / Boss in the room • Cheap shots • Time commitment / Scheduling 	<ul style="list-style-type: none"> • Always make the scribe and the facilitator two different people.
Brainstorming	The core purpose is to generate ideas for solving a specific problem. It is not intended to be precise. Key stakeholders are gathered together for a short intensive period, typically 2-3 hours. The ideas are encouraged by a facilitator and instantly recorded by a scribe.	<ul style="list-style-type: none"> • Project sponsor • Facilitator • Scribe • Project manager • IT specialists • Visiting specialists • Users • Business people 	<ul style="list-style-type: none"> • Team building / Encourage buy-in • Instant feedback • Thinking outside the square 	<ul style="list-style-type: none"> • Non-contributors / Shy / Politics / Boss in the room • Scheduling 	<ul style="list-style-type: none"> • No criticism or debate • Let your imagination soar! • Generate as many ideas as possible • Mutate and combine ideas • The scribe and facilitator can be the same person
Interviewing	The analyst approaches users individually for in-depth discussions around the problem. It is simple, direct and detailed.	<ul style="list-style-type: none"> • Project sponsor • Analyst • IT specialists • Visiting specialists • Users • Business people 	<ul style="list-style-type: none"> • Can encourage buy-in from the interviewee • In-depth • No silent participants • Relatively easy to schedule 	<ul style="list-style-type: none"> • Gradual discovery of individual biases • Slow 	<ul style="list-style-type: none"> • The analyst must decide the general questions that need to be asked before the interview, but be prepared to ask new questions during the interview. • Research the obvious background beforehand. • Use a questions and answers template. • The scribe and facilitator is usually the same person, but not always.
Surveying	Prepare a document for circulation, to larger numbers of users, which asks very specific questions.	<ul style="list-style-type: none"> • Project sponsor • IT specialists • Visiting specialists • Users • Business people 	<ul style="list-style-type: none"> • Relatively easy to schedule • Cover more people • Corroborate other findings 	<ul style="list-style-type: none"> • The scope is limited to a direct answer to a direct question. • Ambiguity and bias very difficult to overcome 	<ul style="list-style-type: none"> • The questions in the survey must be unambiguous. • Set deadlines for response • More suited to confirming existing ideas, than generating new ones.
Prototyping	Types of prototypes Throwing away Evolutionary Operational Narrow focus Broad focus User interface vs. algorithmic	<ul style="list-style-type: none"> • Project sponsor • Developers • Users 	<ul style="list-style-type: none"> • Encourages buy-in • Exposes requirements errors early • Provides something that users can "Touch and See" early on, allowing users to clarify their thoughts. • Demonstrates progress 	<ul style="list-style-type: none"> • Prototyping death spiral i.e. endless tinkering with the prototype prevents "real" development. • Time consuming. • There is a danger of "Throwaway" prototypes being adopted in place of the real system. • The IT team eagerly jumps on the keyboards to the detriment of requirements analysis. 	

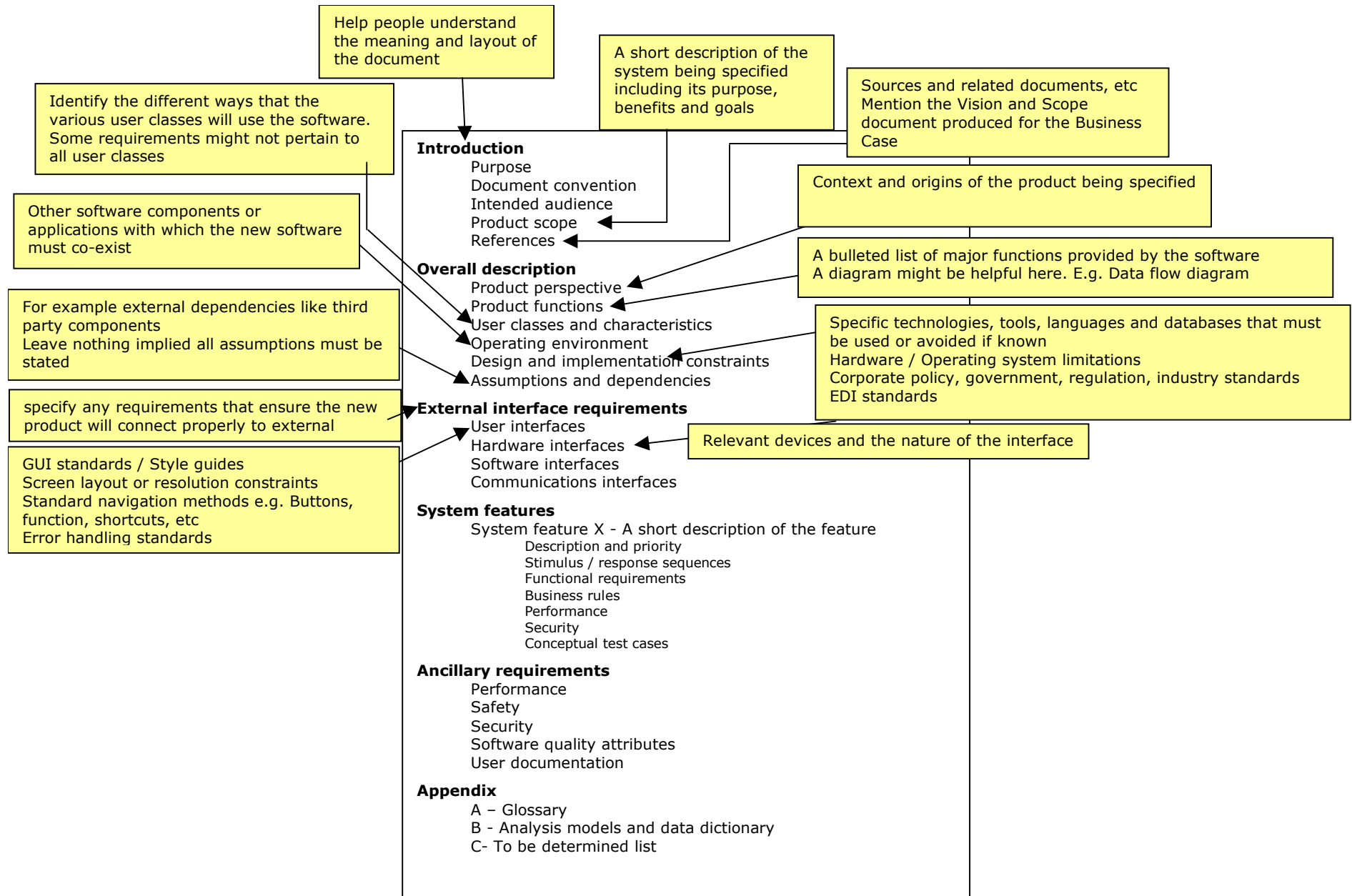
Consolidating What You've Learnt

Your requirements gathering will yield unwieldy amounts of data. Without structure and context, this data will seem overwhelming, and will have little chance of becoming knowledge.

A requirements specification document will help you structure the information you've gathered. It's a practical document which should be used as a blue-print for the development project.

Incidentally, if you prepare a template before you begin gathering, you'll know the sort of information you're going to need in the end. Then you can use your template like a compass – keep referring back to it to make sure you're heading in the right direction.

Here's a Requirements Specification Template you will hopefully find useful...



Verifying Your Requirements

Now that the hardest part's complete, you're 95% done. The last step is to verify your findings. Although you probably feel you couldn't face another round of document reviews, it's important to submit your documentation to all your stakeholders in order to get their feedback. These reviews may take some time, but it's important to remain patient. Remember, the requirements definition process is not finished until your audience says it is.

Conclusion

Remember:

1. The consequences of not defining requirements properly can be crippling
2. Pinpoint the problem to be solved
3. Get the business to buy in
4. Define and communicate the system requirement

Requirements definition is arguably the most essential stage of any software project.

This series has been long, but we hope you've found it informative. Should you have any questions, or would just like some advice or guidance, please give us a call at Infosphere on 1300 558 551.