COM 1030 Requirements documents

Stage 1 of the Crossover involves the preparation of a

Requirements document for your client.

This contains several sections:

Introduction and background;

Elementary business model;

User characteristics;

Functional requirements;

Non-functional requirements;

Dependencies and assumptions;

Constraints;

User interface characteristics;

Plan - a schedule of work with milestones, meetings and deliverables.

Glossary of terms (with an index)

This is a simplified version based on an international standard from IEEE

ALL documents MUST have the following information:

Document name and version - this is very important

document type - e.g. requirements document

author(s) .....

version .....

date .....

We will look at each section shortly.

However, some important points to note.

The document needs to have consistent style and format,

It will be written by the team - usually different people are responsible for different sections.

When the document is assembled from these sections it is best if one person takes responsibility for the editing.

Make sure the style and formatting is consistent.

Make sure that it is properly spelt and grammatically correct.

Section 1, Introduction and background:

This is essentially the short statement that describes the project - it is stuck on the folder.

Section 2, Elementary business model;

This describes the business background, some general statements of company objectives, a description of their market, suppliers, general organisation etc.

Section 3, User characteristics;

Who are the users?

This section describes who they are, what their role in the business is, what their skills are, and any limitations they might have.

Section 4, Functional requirements;

What the system has to do.

Usually these are expressed in terms of statements such as:

"Sales personnel can add new customers to the system"

Essentially functional requirements can be thought about in the following way:

The user has a goal of some sort, often these can be described in terms of:

creating some new piece of data - eg. setting up a new customer

retrieving some existing data: finding out a customer's balance

changing some existing data : updating the customer's account (this includes deleting data),

Such requirements can be stated in English but this can be ambiguous so we have to be careful.

Section 5. Non-functional requirements;

Here we find out about how well the system has to do things.

Non-functional requirements are concerned with specifying and measuring the quality attributes of the system.

We divide non-functional requirements into two categories:

quality attributes, which determine how well the system should perform

and *resource attributes*, which constrain or limit the possible solutions to your business problem.

Unless these are addressed a system may not be successful, even

if all the functional requirements are met.

For most software systems some of these attributes will be *critical*,

that is, unless each of those attributes achieves some required level then the system will probably not be successful,

no matter how well it may meet its functional requirements or meet the goals for its other attributes.

Thus it is essential to identify all the attributes, and to identify which ones are critical, and then to ensure that they are all met.

Identifying Attributes

The International Standards Organisation provides a taxonomy of quality attributes in its draft standard for software systems, (ISO 9126).

As you read through the following list, based on that standard, make a note of those attributes which you feel could be critical to your project.

The list is not exhaustive: you may see other classifications of qualities elsewhere and you may identify critical qualities for your system that do not appear here.

Attributes related to the Functionality:

Suitability - the presence of an appropriate set of functions for specified tasks

Accuracy - the presence of correct and predictable results from specified input

Interoperability - the ability to interact with other specified systems

Compliance - the adherence to specified standards, laws and regulations

Security - the ability to prevent unauthorised access to programs and data

Reliability attributes:

Maturity - the frequency of faults/ rate of software failure

Fault tolerance - the continuity of software execution in the presence of faults

Recoverability - the ease with which performance and data can be recovered in the case of system failure

Usability:

Understandability - the effort required by users to recognise application concepts and their applicability to user tasks

Learnability - the ease with which an application's functions can be learned

Operability - the effort required by users to operate and control the application

From the ISO 9241 standard for usability in software and hardware design a number of other issues are identified such as:

System efficiency:

Time behaviour - the adequacy of system response and performance times

Resource behaviour - the acceptability of amount (and duration) of resources consumed in performing system functions

Maintainability:

Analysability - the ability to identify and diagnose deficiencies in the system

Changeability - the ability to modify the system, to add new functions, remove faults or adapt to environmental change

Stability - the risk of unexpected effects arising from modifications to the system

Testability - the ease with which correct system functioning can be verified

Portability:

Adaptability - the opportunity afforded to adapt the system to different specified environments

Installability - the effort required to install the software in a specified environment

Replaceability - the effort required to use the software in place of other software in a specified environment

This list of attributes is much larger than you will require.

Select the most appropriate and concentrate on these.

Usually for the Crossover project the key ones are: **reliability, usability** and **efficiency**.

Specifying the acceptable level of an attribute.

Having identified critical quality attributes, you need to specify what level or measure of each attribute is *acceptable* in your system.

You should identify at least:

the *worst* acceptable level

the *planned* level - be ambitious, but remain realistic!

the *best* level - just to provide a marker for what might be technically possible but infeasible for you.

Naturally the best comes at a price and so there may be trade offs needed between quality and cost.

It is not good enough to specify that your system will be "very" efficient, “easy to use” or "extremely" adaptable.

You must attempt to define *operational*, *measurable* criteria against which your system can be judged.

This will lead on to defining a set of tests that will establish whether the attribute has been delivered to the required level.

We will look at testing in later but it will often be important to identify, at least in general terms, what the testing approach will be.

For example, if one of your usability criteria for your system is its suitability for the task,

a measure of its *effectiveness* is the *percentage of user goals achieved in a given time*;

a measure of its *efficiency* is the *time for a type of user to complete a set of tasks*;

A series of experiments (tests) could be organised in which users are asked to carry out some important tasks using the system,

we would then be measuring how well these were carried out, how long it took, how many mistakes were made etc.

These experiments should be repeated with as many people as possible in order to get a useful result.

Alternatively, a measure of *satisfaction,* for example, can be gained on a *rating* scale, e.g. a scale of 1-5 by using suitable questionnaires distributed to a selection of users during a trial period of evaluation.

If access to real users is not possible in the timescale you could use some of your friends, preferably those with a similar knowledge of computing as the intended users of the system.

For each numbered attribute we will specify a quality level and eventually a test for determining whether it is met in the final, delivered, software.

Dependencies and assumptions;

The system may have to interact with another and this will impose a number of restrictions on what can be done.

The client may require the system to be implemented for a particular operational environment - operating system, etc.

Constraints;

These could be time constraints or financial constraints etc.

User interface characteristics;

Any special issues - technology to use - mouse, touch screen,

look and feel etc.

Plan - a schedule of work with milestones, meetings and deliver-ables.

This needs to be as realistic as possible.

It is notoriously difficult to estimate how long - and thus how much - a software development project will take.

Glossary of terms (with an index)

Must be included.

Industrial talks

Wednesday October 18th at 2.00 pm

in Stephenson Lecture Theatre 1

Speakers from:

Accenture

Deloitte