Characterisitics of Good Requirements

Requirements Analysis

Requirements analysis is a tool that can be used to ensure that designers capture all the whole-of-life needs of the product or system from the perspectives of all the stakeholders - the **acquirer**, the **operator**, the **user**, the **maintainer** and those who refurbish or dispose of the it at the end of its life. At the end of the process, the designers are left with a **document** listing the system requirements for a product design. From this document they can base much of their subsequent work.

The advantage of carrying out a requirements analysis over a more informal approach is that it enables the designers, users and owners of a product to clarify what the important aspects of the design are. The process has a number of advantages,namely:

- Misunderstandings between the parties over the design requirements is eliminated,
- Conflicting design constraints can be identified,
- The cost of the design is reduced, since unnecessary features can be omitted from the requirements list.

The process of requirements analysis can be undertaken in a variety of ways. According to Stevens [1998], it involves two phases:

1. Identifying user requirements, and
2. Determining system requirements.

Identification of *user requirements* involves intense interaction between the designers and product users. From this process the designers determine the requirements necessary to enable their product to function fully and efficiently.

Having identified the user requirements, the designer then consults the product owner and determines the *system requirements*. This involves analyzing the user requirements and any preliminary specifications. The result is a **document** that lists all of the system requirements. This document is then presented to the customer for review, and conflicts are identified and resolved.

The end result is a document that contains a complete description of all the necessary requirements for a product design.

Characteristics of Good Requirements

As described above, a list of system requirements contains a complete description of the important requirements for a product design. From this list, subsequent design decisions can be based. Of course, if one is to place their trust in them, we must assume that all the requirements in such a list are good. This raises the question,

'How does one differentiate between good requirements and those that are not so good?'

It turns out that good requirements have the following essential qualities:

1. **A good requirement contains one idea.** If a requirement is found to contain more than one idea then it should be broken into two or more new requirements.
2. **A good requirement is clear**; that is, the idea contained within it is not open to interpretation. If any aspects of a requirement are open to interpretation then the designer should consult the relevant parties and clarify the statement.
3. **Requirements should remain as general as possible.** This ensures that the scope of the design is not unnecessarily limited.
4. **A good requirement is easily verifiable,** that is, at the end of the design process it is possible to check whether the requirement has been met. These criteria apply to both user and systems requirements. Examples are given in the section below.

In addition to the qualities listed above, a set of good requirements should completely describe all aspects relevant to a product's design.

Examples

User Requirement

A good user requirement is listed below [Stevens,1998],
The seat shall be comfortable for 95% of the population of each country in which the vehicle is sold.
It meets the three criteria as follows:
1. **It contains one idea**. If the requirement had also made reference to leg room, then the requirement would need to have been broken into two requirements.
2. **It is easily verifiable**. The statement has given quantitative limits from which the seat can be designed. Stating that, '*The seat shall be comfortable for the majority of its indended users,*' would be unnacceptable.
3. **The requirement is general.** The requirement does not state how the seat should be made so as to be comfortable for 95% of the population. To do so would unnecessarily narrow the scope, and limit the design process.
4. **The requirement is verifiable.** To test whether the seat is comfortable or uncomfortable for the right number of people, you need only get people to sit in it, and see if during normal operation they experience discomfort.

System Requirement

Below is a systems requirement that could be improved:
All software shall be written in the C language
This statement is **too specific.** If we were to assume that the software was being written for implementation into an existing system, and that the existing system was written in the C language, then we can gain some understanding of what is being asked for. In fact, the statement may seem reasonable. However, in being specific, the designers' scope has been limited. It may be possible to design the new software to be compatible with the old system, whilst not using C. Such a possibility should be considered.
Furthermore, if we were to quiz the customer further, we might discover that their underlying desires were that all *new* software should be written in the same language for simplicity, and that this software should be compatible with the existing system. We might discover that they do not mind whether or not the new software is designed using C or not. The customer had just stated it that way in their technical specification because **they did not know that the other option existed**. In instances such as this, requirements analysis demonstrates its worth.
Armed with our new knowledge of what the customer really wants, our original design requirement can be split into two newer, but more general requirements:
1. *All new software shall be written in the same language*,
2. *All new software shall be compatible with the existing, C based, system.*

References

Stevens, R., 1998. ***Systems Engineering, coping with complexity***, London, Prentice Hall Europe. (Call Number:
TA168.S97)
Possible further reading (access temporarily unavailable):
DSMC (1999) ***Systems Engineering Fundamentals***, Defence Systems Management College Press, Fort Belvoir, VA or
http://www.dsmc.dsm.mil/pubs/gdbks/sys_eng_fund.htm
International Committee on Systems Engineering (INCOSE), Requirements Working Group has a range of documents on
requiremenst analysis http://www.incose.org/rwg/writing.html
Jonathan Mullins

SUMMARY OF DOCUMENTATION for PRODUCT DESIGN

General Requirements
The general specifications of the operation of a product that must be met when by the end product. These specifications should define the characteristics of the product as experienced by the end user. Examples might include such topics as number of users, power consumption, size, weight, and speed. Numerical values should be assigned to a requirement whenever possible. The Acceptance Testing should verify that the product meets the General Requirements.

Detailed Requirements
1. Description of purpose and general operation including a physical description.
2. In normal operation for input and output
   Number and type of inputs (i.e. analog inputs, digital inputs, etc.-What do they represent?
   Range and resolution of values (i.e. 0-600volts, +- .01 volts)
   Frequency range (if required)
      Such values determine the following:
   Samples per second per channel for analog inputs or data rates for digital data
   For outputs, what will the user see (or hear)? How does the user control the product- keyboard, touch screen, etc.
3. Other conditions
   Response to over-range or errors of input values- What happens when something goes wrong?
   Alarm conditions (if necessary)
   "Hard real-time" timing constraints (if any) and overall timing diagram
4. Acceptance Testing Protocol
   What will satisfy the "user" that the product performs as described?
5. Special Requirements
   Power/safety/environmental considerations

Functional Specifications
1. Block diagram and description of software (SW) and hardware (HW) modules used to meet the Detailed Requirements
2. Description of the interfaces between modules – type of data exchanged, data rates, error conditions, etc.
3. Timing diagram for critical parts of system
4. More detailed description of the output data or signals
5. For the software modules and data, estimate the storage requirements.
5. Details of the User interface with the product

Detailed Design (Hardware and Software)
1. Flow charts, circuit diagrams, etc that describe how the product will be made.
2. Define all the I/O drivers and interrupt service routines. How will these routines be tested?
3. How will the integrated system (HW and SW) be tested?

General (and Detailed) Requirements for an Instrument	
Who is the user?	Engineer or lay person
What is the purpose and general operation of the product?	Examples: Measurement of analog values, storage, display, control, etc.
Are there constraints?	Power/Safety/Environment/Size/Weight/Cost/ Special timing requirements?
What are the general interfaces to the outside world?	**Inputs** (Analog, digital, switches, keypad ?) **Outputs** (Analog, digital ?) **Communication** (To PC/ other instruments?) **User inputs and responses:** What can the user choose? What will the user see or hear?
Other Questions of a general nature.	Are there constraints on manufacturing/ Is updating necessary/ Maintenance? What is the expected production volume - several units / large volume -a consumer product?
More Technical Questions **INPUTS**	**Inputs:** Number and type of inputs (Analog, etc) Range and resolution of values (i.e. 0-600volts +- 0.1 volts.) Frequency range (if it applies)
PROCESSING	Conversion to engineering units / Analysis ?
ERROR CONDITIONS	What are the error conditions and what is the action to be taken ? (i.e. Reset the system/ Ignore ?)
OUTPUTS	Type of display; number of digits; Update rate
STORAGE	What values are to be stored? / How many? What format?/ How retreived
ALARMS	Alarm for over-range or bad data?
ACCEPTANCE TEST	What satisfies the user (or test users) that the product works as required?

NOTE: The emphasis in the General Requirements is on the operation of the product and its relation to the **OUTSIDE WORLD** - not the requirements for the components, etc and the internal workings - unless a requirement specifies particulars of the internal design.

	Timing between modules - are some modules critical as to timing?
Conversions and Processing	**Input:** Conversion from counts to engineering units. Error checking. Exactly what processing or analysis of data is required? Define output conversion - to ASCII, etc? Define output formats for alarms or errors.
Other functional specification.	Are there constraints that will determine the enclosure, power dissipation, type of display, etc.

NOTE: The emphasis in the Functional Requirements is on the operation of the product and the modules (not necessarily the components) defined by trade-offs between hardware and software. For example, to meet the performance requirements, is hardware needed or will software modules or routines be sufficient?

It is critical to define the INTERFACES between modules and those with the outside world, timing constraints, and more details about the input and output signals. An estimate of the storage requirements and the program size are important.

From this document, a competent designer should be able to select particular components, enclosures, etc. to prototype the product.

Functional Requirements for an Instrument	
Block Diagram	Block diagram and description of Software and Hardware to meet the General or Detailed Requirements. 1.A hardware block diagram showing the interfaces (or ports). 2. A software flowchart and/or state diagram showing the program flow. 3. A timing diagram of the overall operation of the unit.
Define the interfaces	For each interface: What type of data are exchanged? **Inputs** Number of bits for ADC. Updated every T seconds. **Outputs** Format to display and update rate. **Communication** Format, Baud rate, etc. **User inputs and responses:** What can the user choose? What will the user see or hear?
Storage	Are data temporary or to be saved in case of power failure? Define the data to be stored from the inputs: Format and size Number of values Time stamp? (Similarly for the output data.) Estimate storage requirement and draw preliminary memory map.
Timing	Define overall timing for the unit - the main timing cycle. For each interface: Data rates between modules. **Input:** How often is ADC read? How often is User input device scanned? **Output:** (Similarly for each output device)