Introduction to Requirements Engineering

- By way of an introduction into the assignment and core concepts required for this terms work, we need to examine the idea of requirements engineering.

- As an essential part of any computing project, formulating a comprehensive understanding of the problem that is to be solved and the required characteristics of the solution is essential. The material presented in this lecture should provide a good introduction into this underpinning concept.

What is Requirements Engineering?

- Alan Davis defines requirements engineering as

"All activities up to but not including the decomposition of software into its actual architectural components" [Davis, 1988] (i.e. the internal system design)

- Bray expands upon this definition:-

"An investigation and description of the problem domain and requirements. Followed by the design and documentation of the characteristics for a solution system that will meet those requirements" [Bray, 2002]

- Kovitz provides a similar definition when he describes requirements engineering as "the process of converting an open-ended problem into a well defined problem" with an appropriate solution.[Kovitz, 1999]

- It should be clear from each of these definitions that the process of requirements engineering involves significantly more than simply formulating a list of required system functionality for a proposed application solution.

Why is Requirements Engineering so Important?

- Appropriate requirements engineering is fundamental to the success of any software engineering project because it provides the foundation upon which the rest of the project is built.

- Inaccuracies, errors and omissions during this pivotal stage can result in problems that can significantly increase the cost and duration of the project.

- A project designed and built around an incorrect understanding of the original problem domain, will result in the production of a system that does not fulfil the needs of the planned user group or client.

- Examples:-

1) Performing Rights Society. (PROMS) Project. This project was abandoned in 1992 after £11 million had been invested. Poor requirements engineering was the

key factor. It was reported that the system requirements were produced in such a way that the user group could not understand or check the planned functionality.

2) London Stock Exchange TAURUS Project. Cancelled in 1993 with an estimated total failure cost of £480 million. It was determined that failure to reconcile conflicting system requirements was at the root of the project breakdown.

3) London Ambulance Service Despatch System. Closed down in 1992 after two days of operation. Sutcliffe identified the major problem with the project as a poor requirements analysis.

- Glass comes to the following conclusion:-

"There is little doubt that project requirements are the single biggest cause of trouble on the software project front. Study after study has found that, where there is a failure, requirements problems are usually at the heart of the matter." [Glass, 1998]

Why is Requirements Engineering so Difficult?

- We can assume, in its simplest form, that requirements engineering is about solving problems.

- Logically, this implies that in order to solve the problem appropriately we must first fully understand what the problem is.

- Thus the task of requirements engineering is doubly difficult, as the engineer needs to:-

  ■ Gather the domain of knowledge from a variety of sources, for any given system and superimpose this upon the expertise required for the requirements engineering itself.

- As the best source of domain knowledge will usually be the existing or potential system users, this also introduces the difficulties of communication and knowledge transfer.

Basic Terminology and Concepts

- In order to fully understand the requirements engineering process we need to be familiar with some basic terminology and concepts:-

  ■ **The problem domain**

  - The problem domain can be defined as the part of the universe within which the problems exist.

  - It can equally be regarded as the part of the world within which the new system solution (or application) will operate in order to produce the required effects.

Example Problem: ***"We want a system to increase the effectiveness of our current stock control system, used to monitor the flow of stock coming in and out of our warehouse"***

- The problem domain would include, to name but a few: -

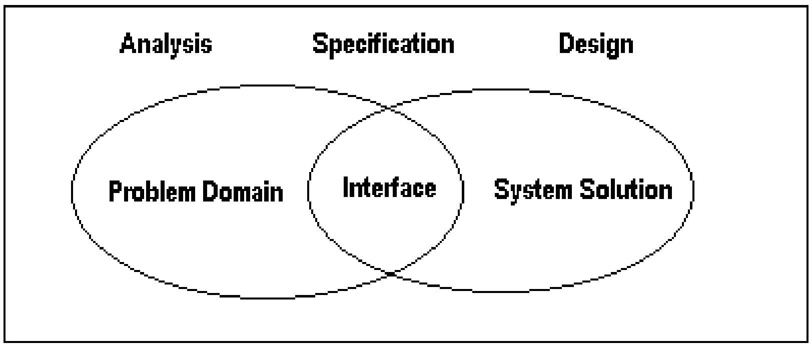
  a) The current stock control system.
  b) The employees/job responsibilities/roles involved in the process.
  c) The clerical documents used/required during the process.
  d) The process inputs and outputs and any associated procedures, including stock suppliers, transport methods and delivery schedules.
  e) Comparable stock control software systems available on the market
  f) Any relevant legislation related to the stock control process or the proposed software solution system.

- A problem domain can sometimes be regarded as a set of sub-systems, known as terminators, which will eventually interface with the new proposed solution system.

■ **The Solution System**

- The solution system, is the system that will eventually be built to solve the problem(s) within the problem domain. (Also called the application or product)

- The problem domain and the solution system are interfaced by the system specification.



- Where: -

■ **Analysis** concerns the problem domain and the problems that exist within it.
■ **Specification** concerns the interaction between the problem and the system solution.
■ **Design** (Not a part of the requirements engineering process) concerns the internal workings of the solution.

■ Requirements

- The following definition of system requirements stresses the difference between the problem domain and the solution system.

  “Requirements are the effects that the client wishes to be brought about in the problem domain.” [Bray, 2002]

- An alternative definition by the IEEE also gives additional insight into the purpose of system requirements:-

  “A condition or capability that must be met by the system to solve a problem or achieve an objective”

- System solution requirements should include four key elements:-

1) Functional Requirements

- Functional requirements describe the required behaviour (functionality) on the part of the solution system.

2) Performance Requirements

- Performance requirements may be regarded as parameters of functionality in that they determine how quickly, how reliably, etc. functions must operate.

- Typical performance requirements include a measure of :-

■ Speed

This can be viewed in terms of throughput or response times. (E.g. The time taken to process 100 transactions)

■ Capacity

Gives an indication of the quantity of data that can be stored within the system and/or the number of operations that can be performed concurrently. (E.g. The number of users that can access an online system simultaneously or the maximum number of data records the system can store)

■ Reliability

The mean time between failure or an indication of system availability. (E.g. An online system will be available for 165 hours, out of a possible 168 hours each week)

■ Usability

Usability evaluation criteria are sometimes difficult to define, but they should give some indication of how 'easy' the system is to operate. (E.g. An experienced computer user should be able to learn how to input records into the system, at a rate of 20 per hour, with no more than an hours training)

3) Design Constraints

- Design constraints are the non-functional requirements that affect (constrain) how the system is built but NOT what the system does.

- Clients (not the requirements engineer) should impose design constraints

- Common design constraints include:-

  - Target operation system.
  - Distributed or local architecture.
  - Available memory size.
  - Front-end graphic styles. (e.g. consistent company format)
  - Programming languages to be used.
  - Application packages that must be integrated with the proposed solution system
  - Development standards that must be applied.
  - Design approaches that must be used.

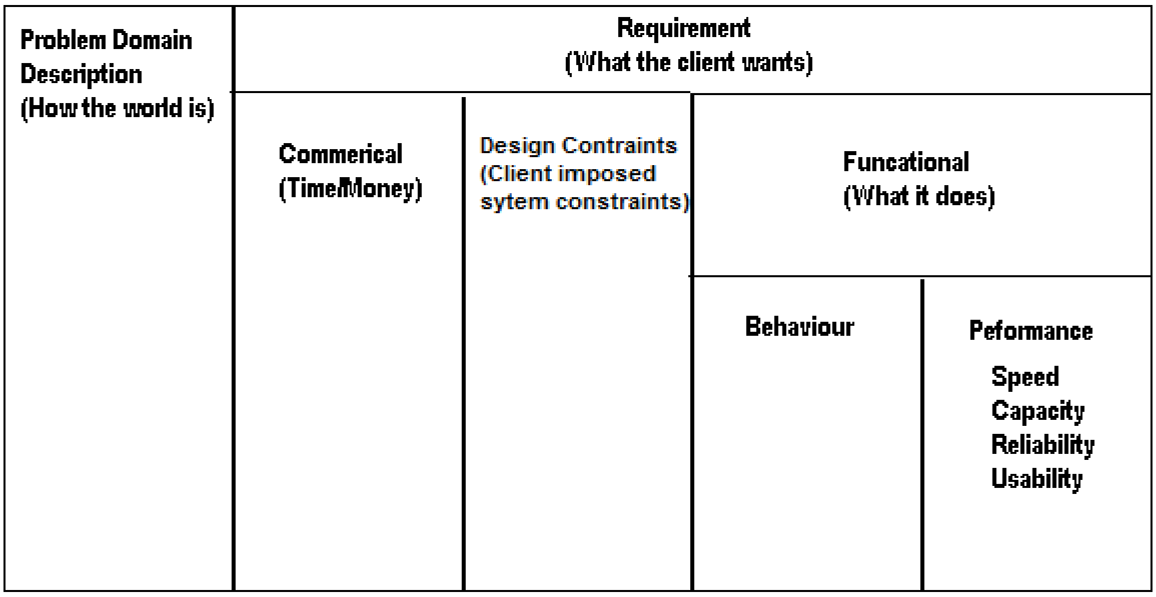
4) Commercial Constraints

- Considerations such as the cost and project time scales must also be considered.

- A variety of project management techniques have been developed to support potentially difficult cost/time scale estimations.

Figure 1.1 (below) summaries the fundamental sections to be included in a typical system solution requirement documentation.

Fig 1.1 - The fundamental sections to be included in any system solution requirement documentation


Problem Domain Description (How the world is)
Requirement (What the client wants)
Commerical (Time/Money)
Design Contraints (Client imposed sytem constraints)
Funcational (What it does)
Behaviour
Peformance
Speed
Capacity
Reliability
Usability

References

Bray, I. K. (2002). An Introduction to Requirements Engineering. Harlow, London, New York, Addison-Wesley.

Davis, A. (1988). A comparison of techniques for the specification of external system behaviour. *Communications of the ACM, 31 (9),pp* 1098-1115.

Glass, R. L et al (1992), Software Tasks - Intellectual or Clerical. *Information & Management (23:4) Oct 1992, pp 183-191*

Kovitz. B. L. (1999), Practical Software Requirements: A Manual of Content and Style. Manning Publications Company