Software Engineering: Requirements, Analysis and Design

Lecture Outline

- Requirements Engineering
- Intro to Analysis and Design
  - Defining Analysis and Design
  - Why do Analysis and Design?
  - Types of Analysis and Design
Challenge facing system and software engineers:

- How can we ensure that we have specified a system that:
  - Properly meets the customer’s needs
  - Satisfies the customer’s expectations

Requirements engineering provides mechanisms for:

- Understanding what the customer wants
- Analysing need
- Assessing feasibility
- Negotiating a reasonable solution
- Specifying the solution
- Validating the specification
- Managing the transformation of the requirements into an operational system
The requirements engineering process can be described in six steps:

- Requirements elicitation
- Requirements analysis and negotiation
- Requirements specification
- System modelling
- Requirements validation
- Requirements management

We will discuss some of these in more detail.

What is Requirements Engineering?

The process of finding, analysing, documenting and checking services and constraints

What is a Requirement?

A range of definitions depending who’s talking!

Manager

An Abstract description of Services or Constraints for System

Coder

Detailed formal mathematical definition of a system function
So where do all these different requirement definitions come from? *Abstraction* (Davis)

“If a company wishes to let a contract for a large software development project, it must define its needs in a sufficiently abstract way that a solution is not pre-defined. The requirements must be written so that several contractors can bid for the contract, offering, perhaps, different ways of meeting the client organisation’s needs. Once a contract has been awarded, the contractor must write a system definition for the client in more detail so that the client understands and can validate what the software will do. Both of these documents may be called the requirements document for the system.”

Say that again ....
What is a requirement?

✦ It may range from
  ✦ a high-level abstract description of a service or of a system constraint
  ✦ to a detailed mathematical functional specification
✦ This is inevitable as requirements may serve a dual function
  ✦ May be the basis for a bid for a contract - therefore must be open to interpretation
  ✦ May be the basis for the contract itself - therefore must be defined in detail
✦ Both these statements may be called requirements
Importance of ‘correct’ Requirements

It might seem that this should be simple:

- Ask the customer, users, and others:
  - What the objectives for the system are
  - What is to be accomplished
  - How the system fits into the needs of the business
  - How the system will be used on a day-to-day basis

In fact it is hard!

- Problems of scope
- Problems of understanding
- Problems of volatility

Reasons for failure of problems with software development:

- Incomplete requirements: 13.1%
- Lack of user involvement: 12.4%
- Lack of resources: 10.6%
- Unrealistic expectations: 9.9%
- Lack of executive support: 9.3%
- Changing requirements and specifications: 8.7%
- Lack of planning: 8.1%
- System no longer needed: 7.5%

Good requirements capture helps reduce the cost of software development.

Requirements capture is involved in a majority of failures!
Why its Difficult

Our aim is to build the right system

- a system that meets the users’ needs

- requires the collaboration of several groups of participants with different backgrounds

Knowledge GAP

CHALLENGE: How to bridge this gap?

The software engineer needs to:
- learn about application domain and “discover” requirements
- choose an appropriate representation for specifying requirements
- May need to “educate” user

Problems of Scope

- The boundary of the system may be ill-defined
  - Who is going to interact with the system?
  - What other systems are involved?
  - Exactly what functionality is the responsibility of the system
    » e.g. should a rostering system produce a telephone directory?

- The customer/user may specify unnecessary technical detail that may confuse overall system objectives
  - e.g. specifying OS, language, hardware, etc. for no particularly good reason
Problems of Understanding /1

✦ Customers/users may:

✧ Not be completely sure of what is needed, e.g.:
  » “See what you can do to help us”
    (Marketing director of textile business)
  » “Try to improve the project”
    (Director of British Aircraft Corporation, with reference to the Concorde project)

✧ Have a poor understanding of the capabilities and limitations of their computing equipment

✧ Not have a full understanding of the problem domain

✧ Have trouble communicating needs to system engineer

✧ Omit information believed to be “obvious”

✧ Specify requirements that conflict with needs of others

✧ Specify requirements that are ambiguous or untestable

Problems of Understanding /2

● collect data on application domain (may include existing system)
  investigation → existing documentation
  observation → work practices
  interviews → questionnaires, personal
  prototyping → interface, functions

● challenge client’s/users’ model of the world
  people: a) filter out, b) distort, and/or c) generalize information
  e.g., challenge statements containing universal quantifiers such as, all, everyone, always, never, nobody, none!

● elicit problems, not solutions
  Not this: “We need to put in a teleconferencing network among our offices.”
  But this: “Our offices are not communicating in a timely manner.”

● distinguish needs from wants; prioritize needs
  needs: features critical to the system’s success
  wants: features nice to have, but not essential
Problems of Volatility

◆ Requirements change over time
◆ Change is inevitable in most systems, due to factors such as:
  ◆ Changes in customer organization, e.g. 
    » new divisions, new products
  ◆ Changes in scale, e.g. 
    » Number of transactions per day 
    » Number of users 
    » Increased connection bandwidth
  ◆ External changes 
    » Changes in law (e.g. taxation) 
    » Changes in international standards (e.g. MPEG)
  ◆ Customers get new ideas as they become aware of system possibilities

Sommerville and Sawyer give guidelines for addressing these problems
◆ Assess business and technical feasibility for the proposed system
◆ Identify the people who will help to specify requirements, and understand their organizational bias
◆ Define the technical environment in which the system will be placed:
  » Computing architecture, operating system, telecommunications needs
◆ Identify “domain constraints” that limit system functionality or performance
  » Characteristics of business environment specific to application domain
Overcoming Requirements Elicitation Problems (cont.)

- Define one or more requirements elicitation methods
  - Interviews, focus groups, team meetings
- Ensure that many people participate so that requirements are defined from different points of view
  - Identify and record the rationale for each requirement
- Identify ambiguous requirements as candidates for prototyping
  - A means of addressing volatility
- Create usage scenarios to help customers/users better identify key requirements

Products of Requirements Elicitation Process

- The requirements elicitation process will produce work products to be included in the software configuration
  - statement of need and feasibility
  - bounded statement of scope for the system
  - list of customers, users and other stakeholders who participated in requirements elicitation
  - description of system’s technical environment
  - list of requirements and their domain constraints
    - preferably organized by function
  - set of usage scenarios (under different operating conditions)
  - any prototypes that were developed
- All work products are reviewed by participants in requirements elicitation
The products of requirements elicitation form the basis for requirements analysis.

Requirements analysis:

- Categorises requirements
  » Organises them into related sub-sets
- Explores relationships between requirements
- Examines requirements for
  » Consistency
  » Omissions
  » Ambiguity
- Prioritises requirements based on customer/user needs
  » May lead to plan for incremental development

Requirements Analysis Questions:

- Is each requirement consistent with the overall objective for the system?
- Have all requirements been specified at the proper level of abstraction?
  » i.e. not too much technical detail, or exclusion of future possibilities
- Is each requirement really necessary?
  » Or is it an add-on not needed for core system objective?
- Is each requirement bounded an unambiguous?
- Does each requirement have an attribution?
  » Who or what is the source of the requirement?
- Are there any conflicting requirements?
- Is each requirement technically achievable in specified environment?
- Is each requirement testable once implemented?
Requirements Negotiation

◆ It is common for customers/users to ask for more than can be achieved
◆ Also common for different stakeholders to proposed conflicting requirements
  ◆ e.g. cost requirement from management vs. performance requirement from users
  ◆ Each party might argue that their requirement is “essential”
◆ Requirements engineer must resolve these conflicts through negotiation:
  ◆ Prioritise requirements, discuss conflicts in this order
  ◆ Identify risks associated with requirements
  ◆ Produce “guesstimates” of development effort for each requirement
◆ In an iterative process, modify, combine or eliminate requirements so that each stakeholder gets some satisfaction

Requirements Management

◆ We have noted that changes in requirements:
  ◆ are essentially unavoidable
  ◆ will persist throughout the lifetime of the system
◆ Requirements management helps the project team to identify, track and control requirements and changes to them
  ◆ This is closely related to configuration management
◆ Traceability tables are developed for requirements
**Requirement flavours**

- Requirements can be considered to come in two, or three, flavours depending on your perspective.
  - Functional requirements
  - Non-Functional requirements
  - Domain requirements

**Functional and non-functional requirements**

*(an artificial distinction?)*

- **Functional requirements**
  - Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.
  - *FR1: The system should read my id card, then request and read my account password and tell me my account balance.*

- **Non-functional requirements**
  - Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.
  - *NFR1: ID cards should be authenticated within 30ms*

- **Domain requirements**
  - Requirements that come from the application domain of the system and that reflect characteristics of that domain
  - *The ATM should be weatherproof (IP65)*
Functional requirements

- Describe functionality or system services
- Depend on the type of software, expected users and the type of system where the software is used
- Functional user requirements may be high-level statements of what the system should do .... but ...
- Functional system requirements should describe the system services in detail

What services does the system provide?

Requirements imprecision

- Problems arise when requirements are not precisely stated - in fact it's one of software's biggest problems!
- Ambiguous requirements may be interpreted in different ways by developers and users
- Consider the term 'appropriate viewers'
  - User intention -
    » Special purpose viewer for each different document type
  - Developer interpretation
    » Provide a text viewer that shows the contents of the document

What I mean

What you think I mean

Paul Dunne GMIT
Requirements completeness and consistency

- In principle requirements should be both complete and consistent
  - Complete
    » They should include descriptions of all facilities required
  - Consistent
    » There should be no conflicts or contradictions in the descriptions of the system facilities
- In practice, it is impossible to produce a complete and consistent requirements document!

Non-functional requirements

- They may
  - Define (emergent) system properties
    » e.g. reliability, response time and storage requirements.
  - and constraints
    » Constraints are I/O device capability, system representations, etc.
- Process requirements may also be specified:
  - use a particular CASE system
  - use a programming language
  - use Type A development method
  - conform to a quality standard.
- Non-functional requirements may be more critical than functional requirements. If these are not met, the system is useless
Possible metrics for specifying non-functional system properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed</td>
<td>Processed transactions/second</td>
</tr>
<tr>
<td></td>
<td>User/Event response time</td>
</tr>
<tr>
<td></td>
<td>Screen refresh time</td>
</tr>
<tr>
<td>Size</td>
<td>K Bytes</td>
</tr>
<tr>
<td></td>
<td>Number of RAM chips</td>
</tr>
<tr>
<td>Ease of use</td>
<td>Training time</td>
</tr>
<tr>
<td></td>
<td>Number of help frames</td>
</tr>
<tr>
<td>Reliability</td>
<td>Mean time to failure</td>
</tr>
<tr>
<td></td>
<td>Probability of unavailability</td>
</tr>
<tr>
<td></td>
<td>Rate of failure occurrence</td>
</tr>
<tr>
<td></td>
<td>Availability</td>
</tr>
<tr>
<td>Robustness</td>
<td>Time to restart after failure</td>
</tr>
<tr>
<td></td>
<td>Percentage of events causing failure</td>
</tr>
<tr>
<td></td>
<td>Probability of data corruption on failure</td>
</tr>
<tr>
<td>Portability</td>
<td>Percentage of target dependent statements</td>
</tr>
<tr>
<td></td>
<td>Number of target systems</td>
</tr>
</tbody>
</table>

Domain requirements

- Derived from the application domain of the system rather than the specific needs of system user.
- Describe system characteristics and features that reflect the domain.
- May be new functional requirements, constraints on existing requirements or define specific computations
- If domain requirements are not satisfied, the system may be unworkable
Domain requirements problems

◆ Understandability
  ○ Requirements are expressed in the language of the application domain
  ○ This is often not understood by software engineers developing the system

◆ Implicitness
  ○ Domain specialists understand the area so well that they do not think of making the domain requirements explicit

Requirements and design

◆ In principle, requirements should state what the system should do and the design should describe how it does this

◆ In practice, requirements and design are inseparable
  ○ A system architecture may be designed to structure the requirements
  ○ The system may inter-operate with other systems that generate design requirements
  ○ The use of a specific design may be a domain requirement
Roadmap

Core Workflows
Requirements
Analysis
Design
Implementation
Testing

Phases
Inception
Elaboration
Construction
Transition

Iteration
Increments

Analysis and Design
Definitions

- Many different definitions of the difference between analysis and design
- In the past there was common agreement
- Was viewed as follows:

<table>
<thead>
<tr>
<th>Description of the Problem</th>
<th>Description of the Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Essential Model</td>
<td>Implementation Model</td>
</tr>
</tbody>
</table>

Role of the Analyst          Role of the Designer

Structured Analysis and Structured Design

- different tools used to develop system
- often different teams used to do analysis and design
- clear distinction between tasks
- leads to the “Analysis/Design Wall”
Definitions (3)

- With newer development methods, the divide between analysis and design breaks down
  - the concept of seamlessness
  - that an essential model is very difficult to develop without reference to the implementation issues
  - the increasing emphasis on requirements analysis
- Some commentators see only
  - Requirements Analysis
  - High-level Design
  - Low-level Design
- The debate rages on!

Definitions (4)

- Systems Analysis
  - Process of defining a problem, gathering the requirements and developing an analysis model representing those requirements
  - Describing the problem the system means to solve
- Systems Design
  - Process by which the analysis model is transformed into a model which is capable of being implemented
  - Describing the solution to the problem
Why do we need Analysis?

Why not go straight to design and implementation?

The benefits of an analysis model are that it:
- yields a more precise and complete specification of the requirements → removes ambiguities and redundancies
- is described in the developers language → more formal/precise
- structures requirements for easier understanding and maintenance
- is a first cut at design

What do we do with the analysis model?
- keep and maintain it throughout the system’s life cycle
- throw it away after analysis is complete
- don’t build it! (not recommended)

Paul Dunne GMIT

Why do analysis and design?

◆ Add formality
◆ Reduce errors
◆ Improve communication between participants
◆ Get the requirements right!
◆ The cost of fixing analysis and design errors as one moves from the analysis phase to the maintenance phase can increase by a factor of 100
◆ Generate documentation

Paul Dunne GMIT
Cost of Analysis/Design Errors

Source of Errors
Types of Analysis/Design

◆ Top-down Analysis and Design
  ◦ Structured Analysis
  ◦ SADT
◆ Object-Oriented Analysis and Design
  ◦ Object Modeling Technique (OMT)
  ◦ Booch OOA&D
  ◦ Unified Modeling Language (UML)
◆ Data-Driven Analysis and Design
  ◦ Jackson System Design
  ◦ Warnier-Orr Method

References