

Requirements Engineering



Lecture Outline

- Overview of Requirements Engineering
- Types of Requirements
- Techniques for Requirements Elicitation
- Requirements Documentation
- Requirements Management

What is Requirements Engineering?

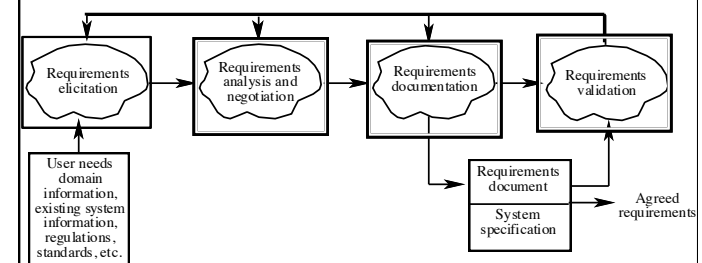
Requirements engineering is the branch of software engineering concerned with the real-world goals for, functions of, and constraints on software systems. It is also concerned with the relationship of these factors to precise specifications of software behaviour, and to their evolution over time and across software families.

[Zave 1997]

Requirements engineering is a systematic process of developing requirements through an iterative, co-operative process of analyzing the problem, documenting the resulting observations in a variety of representation formats and checking the accuracy of the understanding gained.

[Loucopoulos 1995]

Requirements Engineering Process



Agile methods and requirements



- Many agile methods argue that producing detailed system requirements is a waste of time as requirements change so quickly.
- The requirements document is therefore always out of date.
- Agile methods usually use incremental requirements engineering and may express requirements as 'user stories' (discussed in Chapter 3).
- This is practical for business systems but problematic for systems that require pre-delivery analysis (e.g. critical systems) or systems developed by several teams.

Types of Requirements



- **User requirements**
 - Statements in natural language plus diagrams of the services the system provides and its operational constraints. Written for e.g., customers.
 - Example: LIBSYS shall keep track of all data required by copyright licensing agencies in the UK and elsewhere.
- **System requirements**
 - A structured document setting out detailed descriptions of the system's functions, services and operational constraints. Defines what should be implemented so may be part of a contract between client and contractor. Written for e.g., software architect/designers.
 - Example: on making a request for a document from LIBSYS, the requestor shall be presented with a form that records details of the user and the request made.



Types of Requirements (cont.)



- **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.
 - May state what the system should not do.
 - Defines *what* the system is supposed to do
- **Non-functional requirements**
 - Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.
 - Often apply to the system as a whole rather than individual features or services.
 - Not directly related to specific functions.
 - Often related to system performance, usability, security, reliability etc.

Types of Requirements (cont.)



- **Non-functional requirements generally are hard to verify**
 - E.g., LIBSYS should be easy to use
 - E.g., LIBSYS should be highly available
- **These vague goals cause problems to developers**
 - Leave the scope for different interpretations and dispute after system delivery.
- **So non-functional requirements should be quantitative, whenever it is possible.**
 - E.g., A novice user should be able to use all functions of LIBSYS after a total of 5 days' training.
 - E.g., Downtime within normal working hours of LIBSYS shall not exceed five seconds in any one day

Look at These Requirement Examples



- **Requirement 1:**
 - A library system that provides a single interface to a number of databases of articles in different libraries. Users can search for, download and print these articles for personal study.
- **Requirement 2:**
 - The user interface for LIBSYS shall be implemented as simple HTML without frames or Java applets.
 - The system development process and deliverable documents shall conform to the process and deliverables defined in XYZCo-SP-STAN-95.

10

The University of Adelaide

Software Engineering and Project

Lecture Outline



- Overview of Requirement Engineering
- Types of Requirements
- Techniques for Requirements Elicitation
- Requirements Documentation
- Requirements Management

3

The University of Adelaide

Software Engineering and Project

Requirements Elicitation



- Sometimes called requirements discovery.
- Involves technical staff working with customers to find out about the application domain, the services that the system should provide and the system's operational constraints.
- May involve end-users, managers, engineers involved in maintenance, domain experts, trade unions, etc. These are called *stakeholders*.

11

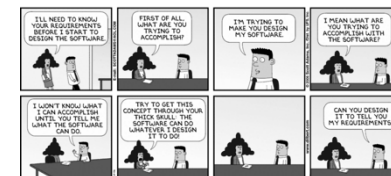
The University of Adelaide

Software Engineering and Project

Problems of Requirement Elicitation



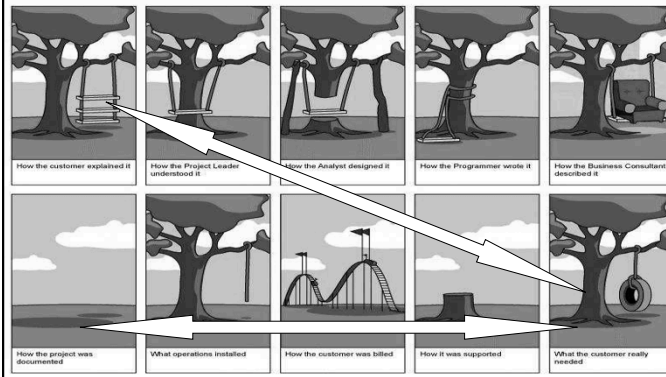
- Stakeholders don't know what they really want.
- Stakeholders express requirements in their own terms.
- Different stakeholders may have conflicting requirements.
- Organisational and political factors may influence the system requirements.
- The requirements change during the analysis process. New stakeholders may emerge and the business environment may change.



12

The University of Adelaide

Why is Understanding Requirements Important?



Main Activities of Requirement Elicitation



- **Requirements discovery**
 - Interacting with stakeholders to discover their requirements.
- **Requirements classification and organisation**
 - Groups related requirements and organises them into coherent clusters.
- **Prioritisation and negotiation**
 - Prioritising requirements and resolving requirements conflicts.
- **Requirements specification**
 - Requirements are documented for system design or next round of requirement elicitation

Requirements Elicitation



- **Techniques for requirements elicitation:**
 - Interviews - the most commonly used technique.
 - Focus group
 - Ethnography - adopted from sociology and includes the observation of users.
 - Contextual query. It is a knowledge acquisition approach.
 - Use cases (Scenarios)
 - Goal-based techniques.
 - Qualify Function Deployment (QFD)
 - Prototype

These techniques can only be used in the later stages of requirements elicitation.

Requirements Elicitation Techniques



Technique 1: Interview



- A technique for eliciting detailed information from stakeholders
- **Type of interview:**
 - Unstructured interviews (open interviews)
 - Ask more open-end questions, no predefined agenda
 - Structured interviews (closed interviews)
 - Ask more specific question (usually predefined) based on the analysis of the results from the unstructured interview – close end questions



Requirements Elicitation Techniques

Technique 1: Interview



- Consists of four phases:
 - Identifying candidates
 - Preparation
 - Arranging the interview
 - Schedule the interview in advance
 - Create a set of goals
 - Set the length of the interview
 - Request or give the interviewee the necessary materials
 - Confirm the interview 1 - 2 days in advance
 - Preparing the questions
 - Context-free product questions (open, close)
 - Context-specific product questions (open, close)
 - Conducting the interview
 - Follow-up

16

The University of Adelaide

Software Engineering and Project

Questions to think about

- What is requirements engineering?
- What are functional requirements?
- What are non-functional requirements?
- What are the techniques for requirements elicitation?
- How to document requirements?
- What activities are involved in requirements engineering?

17

The University of Adelaide

Software Engineering and Project

Requirements Elicitation Techniques

Technique 1: Interview



- **Effective interviewers normally:**
 - Are open-minded, avoid pre-conceived ideas about the requirements and are willing to listen to stakeholders.
 - Prompt the interviewee to get discussions going using a springboard question, a requirements proposal, or by working together on a prototype system.
- **Good:** for getting an overall understanding of what stakeholders do & how they might interact with the system.
- **Not good:** for understanding domain requirements
 - Requirements engineers may not understand domain terminology;
 - Some domain knowledge is so familiar that people find it hard to articulate or think that it isn't worth articulating.

17

The University of Adelaide

Software Engineering and Project

Requirements Elicitation Techniques

Technique 2: Use Case



- Use-cases are a scenario based technique which identify the **interactions** and the actors involved.
- A set of use cases should describe all possible interactions with the system.
- Part of UML
- Sequence diagrams may be used to add detail to use-cases by showing the sequence of event processing in the system.
- Focus on interactions, not effective to elicit constraints/non-functional requirements.

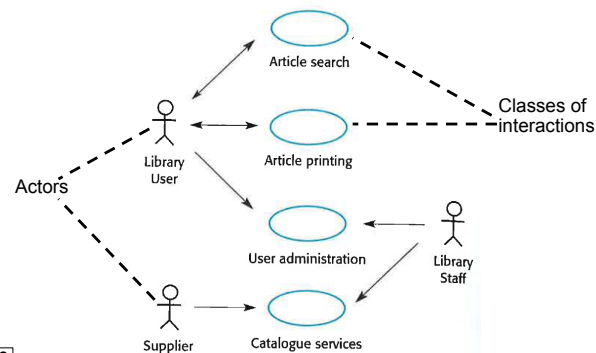
18

The University of Adelaide

Software Engineering and Project

Requirements Elicitation Techniques

Technique 2: Use Case



Requirements Elicitation Techniques

Technique 3: - Prototyping

- A prototype is an initial version of a system which is available early in the development phase
- Prototypes are valuable for requirements elicitation because users can experiment with the system and point out its strengths and weaknesses of the implemented requirements
- Rapid development of prototypes is essential so that they are available early in the elicitation process
 - Some functionality may be left out
 - Non-functional requirements (performance) are less stringent
 - No secondary functions (e.g. maintenance)
 - No complete documentation

Requirements Elicitation Techniques

Technique 3: - Prototyping

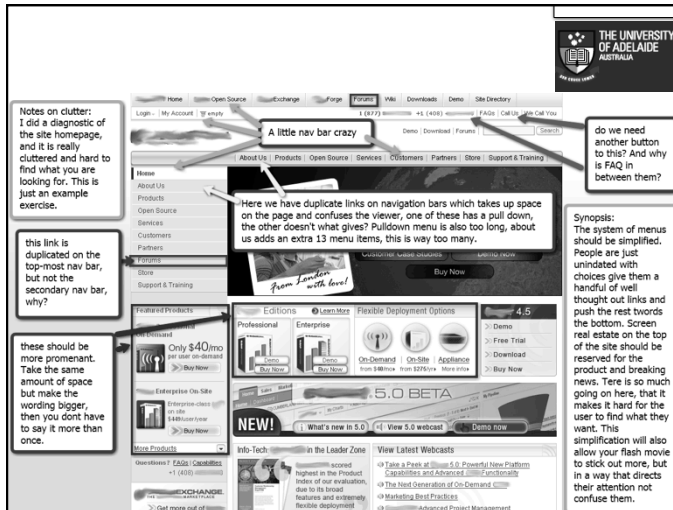
- The prototype allows users to experiment and discover what they really need to support their work; get their feedback
- Establishes feasibility and usefulness before high development costs are incurred
- Essential for developing the 'look and feel' of a **user interface**
 - Probably the only technique to validate interface requirements
- Forces a detailed study of the requirements which reveals inconsistencies and omissions

Requirements Elicitation Techniques

Technique 3: - Prototyping

- Paper Prototyping
 - A paper **mock-up** of the system for feedback
- "Wizard of Oz" Prototyping
 - A person simulates the responses of the system in response to some user inputs
 - http://en.wikipedia.org/wiki/Wizard_of_Oz_experiment
- Executable Prototyping
 - Some rapid development language/environments are used to develop an executable prototype





Notes on clutter: I did a diagnostic of the site homepage, and it is really cluttered and hard to find what you are looking for. This is just an example exercise.

A little nav bar crazy

do we need another button to this? And why is FAQ in between them?

Here we have duplicate links on navigation bars which takes up space on the page and confuses the viewer, one of these has a pull down, the other doesn't what gives? Pulldown menu is also too long, about us adds an extra 13 menu items, this is way too many.

Synopsis: The system of menus should be simplified. People are just undated with choices give them a handful of well thought out links and push the rest towards the bottom. Screen real estate on the top of the site should be reserved for the product and breaking news. There is so much going on here, that it makes it hard for the user to find what they want. This simplification will also allow your flash movie to stick out more, but in a way that directs their attention not confuse them.

this link is duplicated on the top-most nav bar, but not the secondary nav bar, why?

these should be more prominent. Take the same amount of space but make the wording bigger, then you don't have to say it more than once.

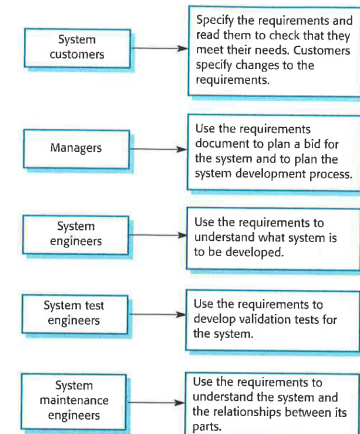
Lecture Outline

- Overview of Requirement Engineering
- Types of Requirements
- Techniques for Requirements Elicitation
- Requirements Documentation
- Requirements Management

Requirements Documentation

- The software requirements document is the official statement of what is required of the system developers.
- The document is normally called Software Requirements Specification (SRS)
- Should include both a definition of user requirements and a specification of the system requirements.
- It is NOT a design document. As far as possible, it should be WHAT the system should do rather than HOW it should do it.

Users of Requirements Document



Characteristics of A Good Requirements Document



- **Unambiguous:** Every requirement stated in the SRS has only one interpretation.
- **Complete:** All the requirements in the SRS are defined, referenced, labelled and compliant with the SRS standard defined.
- **Consistent:** No requirements in the SRS are in conflict with each other.
- **Verifiable:** All the requirements in the SRS are verifiable (testable) against a predefined checklist or international standards.
- **Modifiable:** The structure and style of the SRS are such that any necessary requirements changes can be made easily, completely, and consistently.
- **Traceable:** The origin and rationale of each requirement in the SRS is stated and the links between requirements, design, implementation and test cases are provided.

25

The University of Adelaide

Software Engineering and Project

Characteristics of A Good Requirements Document (cont.)



- The use of "shall" statements is encouraged, as this implies a directive to express what is mandatory.
 - The use of "shall" requires a sense of commitment
- All the requirements shall NOT be defined with ambiguous English
 - "sort of", "kind of", "similar", "more or less" shall NOT be used
 - The use of "will", "may", "should" lead to vagueness
 - "will" statements imply a desire, wish, intent or declaration of purpose.
 - "should" or "may" are used to express non-mandatory provisions.
 - It is recommended that "will, should and may" not be used in writing requirements unless absolutely necessary.

26

The University of Adelaide

Software Engineering and Project

Characteristics of A Good Requirements Document (cont.)



- Invent a standard format and use it for all requirements.
- Use language in a consistent way. Use **shall** for mandatory requirements, **should** for desirable requirements.
- Use text highlighting to identify key parts of the requirement.
- Avoid the use of computer jargon.
- Include an explanation (rationale) of why a requirement is necessary.



27

The University of Adelaide

Software Engineering and Project

Requirements: examples



- **R001 - Register patient**
The PRS shall allow designated staff to register new patients.
- **R002 - Assign ID**
The PRS shall allow the designated staff to assign a unique ID to each patient's record that shall be used throughout their visit in the hospital.
- **R003 - Allocate bed**
The PRS shall allow the designated staff to allocate an available bed to patient.
- **R004 - Personal Information**
The PRS shall store the patients id, first name, last name, doctors name, ward number, bed number, expected visit length.
-

28

The University of Adelaide

Software Engineering and Project

Structure of Requirement Documents



- Requirements documents standards have been designed. These are mostly applicable to the requirements for large systems engineering projects
 - IEEE standard (IEEE/ANSI 830-1998)
 - IEEE830.pdf, available at course website
 - US Department of Defense
- The company you are going to work may have its own standard.

Structure of Requirement Documents



Chapter	Description
Preface	This should define the expected readership of the document and describe its version history, including a rationale for the creation of a new version and a summary of the changes made in each version.
Introduction	This should describe the need for the system. It should briefly describe the system's functions and explain how it will work with other systems. It should also describe how the system fits into the overall business or strategic objectives of the organization commissioning the software.
Glossary	This should define the technical terms used in the document. You should not make assumptions about the experience or expertise of the reader.
User requirements definition	Here, you describe the services provided for the user. The nonfunctional system requirements should also be described in this section. This description may use natural language, diagrams, or other notations that are understandable to customers. Product and process standards that must be followed should be specified.
System architecture	This chapter should present a high-level overview of the anticipated system architecture, showing the distribution of functions across system modules. Architectural components that are reused should be highlighted.

Structure of Requirement Documents



Chapter	Description
System requirements specification	This should describe the functional and nonfunctional requirements in more detail. If necessary, further detail may also be added to the nonfunctional requirements. Interfaces to other systems may be defined.
System models	This might include graphical system models showing the relationships between the system components and the system and its environment. Examples of possible models are object models, data-flow models, or semantic data models.
System evolution	This should describe the fundamental assumptions on which the system is based, and any anticipated changes due to hardware evolution, changing user needs, and so on. This section is useful for system designers as it may help them avoid design decisions that would constrain likely future changes to the system.
Appendices	These should provide detailed, specific information that is related to the application being developed; for example, hardware and database descriptions. Hardware requirements define the minimal and optimal configurations for the system. Database requirements define the logical organization of the data used by the system and the relationships between data.
Index	Several indexes to the document may be included. As well as a normal alphabetic index, there may be an index of diagrams, an index of functions, and so on.

Lecture Outline



- Overview of Requirement Engineering
- Types of Requirements
- Techniques for Requirements Elicitation
- Requirements Documentation
- Requirements Management

Requirements Management



- Requirements management is the process of managing changing requirements during the requirements engineering process and system development.
- New requirements emerge as a system is being developed and after it has gone into use.
- You need to keep track of individual requirements and maintain links between dependent requirements so that you can assess the impact of requirements changes. You need to establish a formal process for making change proposals and linking these to system requirements.
- Changing requirements (at later stages of software development) can be expensive!!!
 - Changing a requirement may cost 10 times more than fixing a bug

Changing Requirements



- The business and technical environment of the system always changes after installation.
 - New hardware may be introduced, it may be necessary to interface the system with other systems, business priorities may change (with consequent changes in the system support required), and new legislation and regulations may be introduced that the system must necessarily abide by.
- The people who pay for a system and the users of that system are rarely the same people.
 - System customers impose requirements because of organizational and budgetary constraints. These may conflict with end-user requirements and, after delivery, new features may have to be added for user support if the system is to meet its goals.

Changing Requirements



- Large systems usually have a diverse user community, with many users having different requirements and priorities that may be conflicting or contradictory.
 - The final system requirements are inevitably a compromise between them and, with experience, it is often discovered that the balance of support given to different users has to be changed.

Requirements Change Management



- Deciding if a requirements change should be accepted
 - *Problem analysis and change specification*
 - During this stage, the problem or the change proposal is analyzed to check that it is valid. This analysis is fed back to the change requestor who may respond with a more specific requirements change proposal, or decide to withdraw the request.
 - *Change analysis and costing*
 - The effect of the proposed change is assessed using traceability information and general knowledge of the system requirements. Once this analysis is completed, a decision is made whether or not to proceed with the requirements change.
 - *Change implementation*
 - The requirements document and, where necessary, the system design and implementation, are modified. Ideally, the document should be organized so that changes can be easily implemented.

Relevance to Your Project



- What techniques to use in order to get requirements?
- How to analyze and document your requirements?

Key points revisited



- Requirements for a software system set out what the system should do and define constraints on its operation and implementation.
- Functional requirements are statements of the services that the system must provide or are descriptions of how some computations must be carried out.
- Non-functional requirements often constrain the system being developed and the development process being used.
- They often relate to the emergent properties of the system and therefore apply to the system as a whole.

Key points revisited (cont.)



- The software requirements document is an agreed statement of the system requirements. It should be organized so that both system customers and software developers can use it.
- The requirements engineering process is an iterative process including requirements elicitation, specification and validation.
- Requirements elicitation and analysis is an iterative process that can be represented as a spiral of activities – requirements discovery, requirements classification and organization, requirements negotiation and requirements documentation.

Key points revisited (cont.)



- You can use a range of techniques for requirements elicitation including interviews, scenarios, use-cases, prototyping, and ethnography.
- Requirements validation is the process of checking the requirements for validity, consistency, completeness, realism and verifiability.
- Business, organizational and technical changes inevitably lead to changes to the requirements for a software system. Requirements management is the process of managing and controlling these changes.