**5SENG007W Software Engineering Principles and Practice**

# Lecture Week 2

Requirements Engineering 1: elicitation methods, types of requirements, formalisation (UML), use case analysis

# Lecture Outline

- Introduction
  - What is software requirement?
  - Why do we need to define the requirements?
  - Types of requirements
- Requirement elicitation techniques
- Requirement prioritisation
- Requirements documentation
- Requirements formalisation - UML

## Role of Requirements

- ‘Requirements’ are needed to define the problem that requires a software development based solution
  - Examples: an old on-line shopping system needs updates; a new gym that needs class/personal trainers booking system; a new structure of the organisation which needs a website (perhaps with different settings for internal and external users)
- Often problems are poorly understood, needs are not clear, a software developer meets ambiguous or conflicting statements – all these to be transformed into well-defined requirements, for which a solution should be developed.
  - Hence, there is a need for a feasibility study – business plan, is it realistic, enough resources, etc

## Role of Requirements

- Building a well-defined set of requirements provides the foundations for a software development project
- So what do requirements do:

*Requirements should define what the system* ***should do*** ***(something needed)****,*

*not how the system should do it*

## *Requirements should define <u>what</u> the system* ***should do***

**Example**: consider a software system which provides a booking functionality for (theatre tickets/cars/medical appointments/etc)

- **Problem**: what the system should do in order to enable users to book a medical appointment?

**Analysis**: what would enable users(patients) to book? Which data are involved? Storage? Registration? Booking a GP or a nurse? Availability calendar: doctors/nurses/rooms

## What is Requirement?

- Requirements define <u>what</u> a system must do for its users and the <u>constraints</u> under which the system must operate (Sommerville, 2015)
- “Requirements describe what the product must do or a <u>quality</u> it must have.” (Robertson and Robertson, 2006)
- A <u>condition</u> or <u>capability</u> that must be met or possessed by a system or system component (IEEE Std, 1990)
- (the references are at the end of the lecture)
- …

## Why bothered?

- Why waste time with defining requirements?
- Can I not start building the system?
- Maybe you can!
  - If the problem is small, simple and very well understood
  - If it won’t cause harm when it fails
- Software development is inherently complex
- It is important to understand the needs well to deliver the right software and avoid risks!

# Requirement Elicitation

- **Requirements elicitation** is the term used for the process of finding and formulating requirements from stakeholders, documents and other sources
- A cooperative process requires a careful study of the problem domain & stakeholder needs
- Major activities in Requirements Elicitation are:
  - **Requirements discovery**
  - **Requirements classification and organisation**
  - **Requirements prioritisation**
  - **Requirements documentation**

## Requirements discovery

*How some executives think requirements are developed.*

# Key Considerations

- Three key considerations for requirements elicitation:
    1. *What information to gather?*
    2. *From which source?*
    3. *By which techniques?*

## 1. Information to Gather

- Progress **from the general to the specifics**
  - Aims and objectives
  - Project scope and constrains
  - User, activities/processes, tasks and aims/goals
  - Software functions/requirements
  - Functional constrains, conditions, restrictions, etc
- Requirements are the end of the elicitation process.
- Intermediate works need delivering, for example
  - *A description of the problem domain*
  - *A list of current issues or problems*
  - *Business processes/activity descriptions*
  - *Business rules or constraints imposed upon the activities, systems etc.*
  - *Risks*
  - *etc.*

## 2. Sources of Information

- Stakeholders
  - Clients, sponsors, customers
  - End-users
  - Others
- Existing systems
- Problem domain experts
- Websites
- User manual or guides
- Law and regulations
- Competitor’s products
- Predecessor products
- Documents defining the characteristics & behaviour of interfacing systems
- Relevant technical standards
- … … …
- … … …

## Stakeholder?

**Stakeholder:**

Anyone who will be affected by the development or implementation of a new system

- A person, group, legal entity
- Interest in the system (positive or negative)
- What are gains or loses from the implementation of system
- What influence the requirements

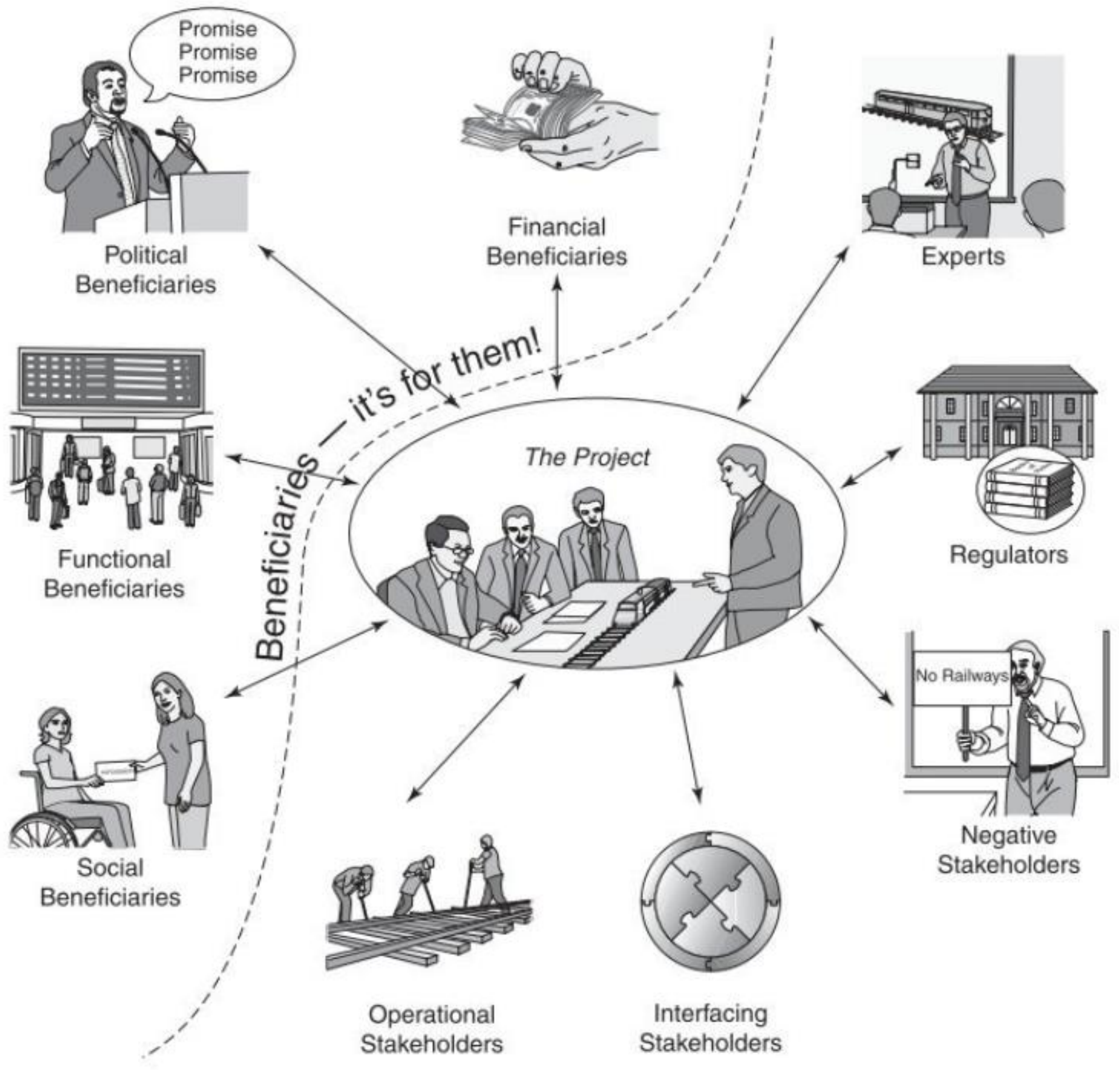


**Figure 2.2:** Beneficiaries and other stakeholders in a transport project.

Source of image: Alexander and Beus-Dukic (2009)

## Why Stakeholder Analysis?

- We need to identify and analyse the stakeholder needs
- They specify requirements – without them requirements are incomplete
- They specify project boundaries i.e. scope, time, budget and so
- They influence the project outcome – powerful stakeholder can affect the project outcome positively
- They may have authority over the software in some way e.g. regulators
- ….

# Identifying Stakeholder

- Need to ask the questions
  - *Who pays for the project?*
  - *Who are existing users?*
  - *Who will be the new users?*
  - *Who will maintain the system?*
  - *Which units or departments will you use the software?*
  - *Will business activities change? If so, who will be affected?*
  - *Will existing positions/roles change by the system implementation?*
  - *Any laws and regulations apply to the owner/project? Who can advise?*
  - *Who has authority to make changes or to stop the project plan?*
  - …..
- A good starting point is Stakeholders Onion Model!

# Onion Model for Stakeholder Analysis

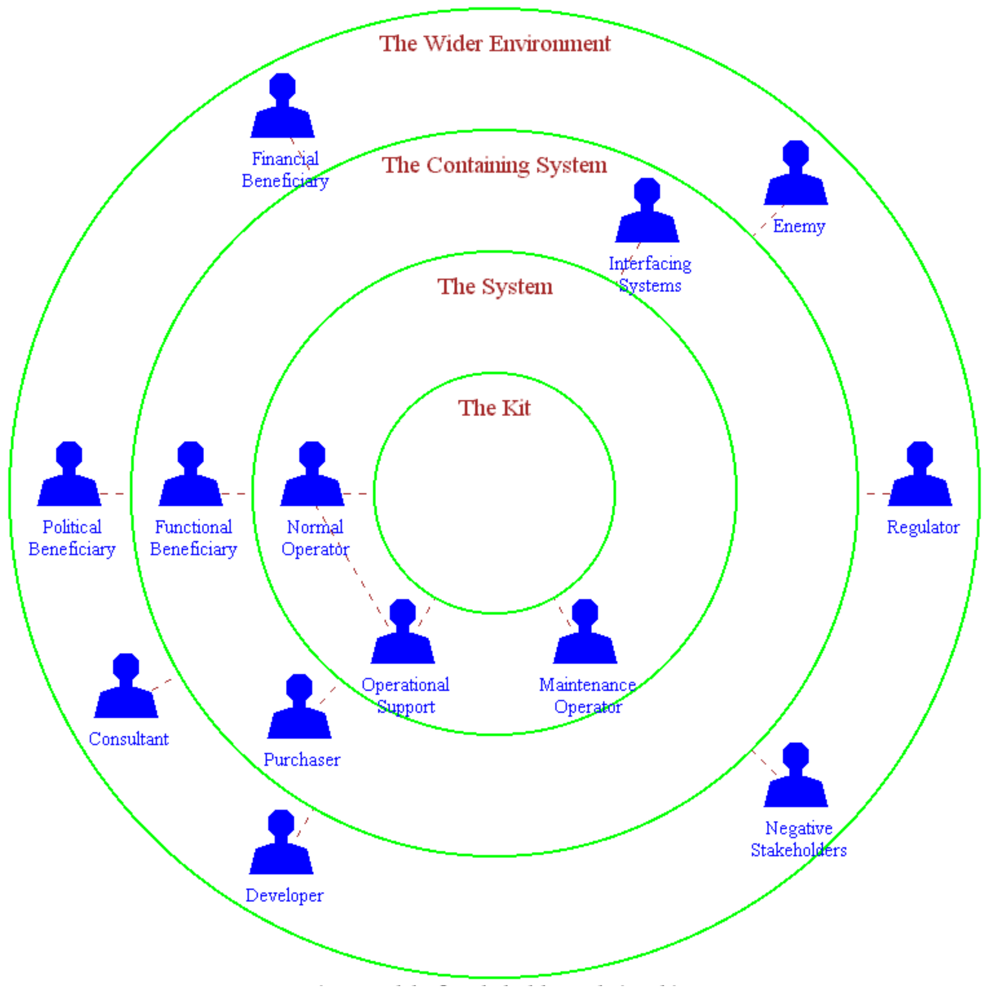
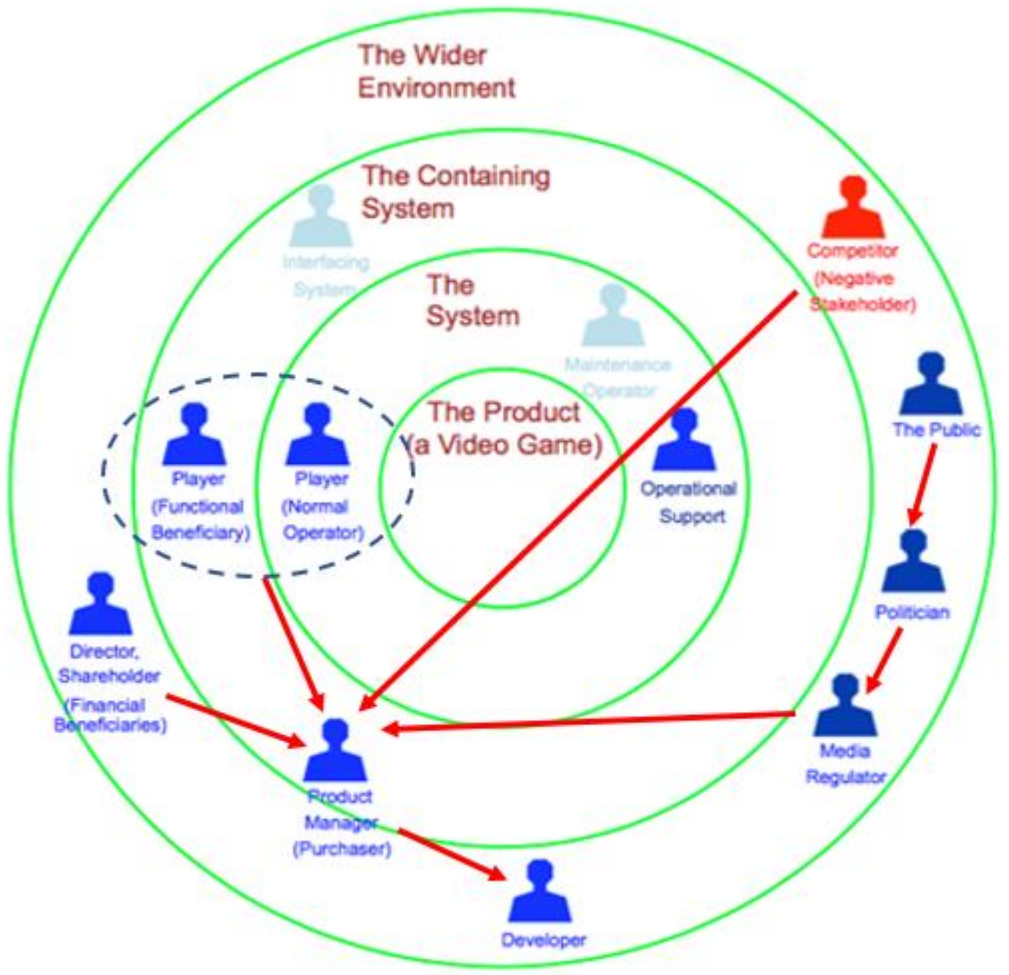


Image source: http://www.scenarioplus.org.uk/

- ❑ Centre circle > Software or system to be delivered
- ❑ SYSTEM
  - All the operational roles
  - They take part in day-to-day operations of the system
  - UML calls operators as actors
- ❑ CONTAINING SYSTEM
  - Beneficiaries who don't interact with the system but receives benefits from its operation for example,
    - Managers
    - Purchaser: who will buy it
- ❑ WIDER ENVIRONMENT
  - Those who are further away but either
    - influence the project outcomes or
    - gain non-functional benefits
  - Regulatory bodies
  - Champion: political or social gain
  - Sponsor: funding
  - Negative: e.g. competitors, environmental groups, etc.

## Example: A Video Game



Source: (Alexander and Beus-Dukic, 2009)

- SYSTEM
  - Operators: player and support people
- CONTAINING SYSTEM
  - Owner: product manager
  - Functional beneficiaries: player, purchaser (e.g. parent)
- WIDER ENVIRONMENT
  - Shareholders as financial beneficiary
  - Competitor as negative stakeholder
  - Public putting pressure on Media Regulator via Politicians
  - Developer - following requirements from Product Managers

# Elicitation Plan: Example

| **Objective**<br>What? | **Technique**<br>How? | **Subject(s)**<br>Who/Where? | **Time needed** |
|---|---|---|---|
| To get background on the company and the advertising industry | Background reading | Company reports, trade journals | 0.5 day |
| To establish business objectives. Agree likely scope of new system. Check out involvement of non-UK offices | Interview | Two directors | $2 \times 1$ hour each |
| To gain understanding of roles of each department. Check out line management and team structure in the Creative department. Agree likely interviewees among staff | Interview | Department heads (only 1 account manager) | $2 \times 1$ hour each |
| To find out how the core business operates | Interview | 1 account manager<br>1 copy writer<br>1 editor | 1.5 hours each |

## 3. Techniques

- Also known as **fact-finding techniques**
  - Background reading
  - Interviewing
  - Group interviews
  - Observation
  - Questionnaire
  - Document sampling
  - Brainstorming
  - Workshops
  - Focus groups
  - Prototyping
  - ... and others

## Background Reading

- Aim is to understand the environment (e.g. organization and its business objectives)
- Includes:
  - Business plan and organization charts,
  - policy manuals,
  - documentation of existing systems,
  - job descriptions
  - etc.
- Appropriate situations:
  - not familiar with the project environment
  - initial stages of fact finding

# Background Reading

- Advantages:
  - helps to understand the environment (e.g. company) before meeting the people who work there
  - helps to prepare for other types of fact finding
  - documentation of existing system may help to identify requirements for functionality of new system
- One major problem or you can say a disadvantage is that:
  - written documents may be out of date or not match the way the organization really operates
  - You may be overloaded with information – try to prioritise

## Interviewing

- The most widely used elicitation technique because
  - personal contact allows the interviewer to respond adaptively to answers;
  - it is possible to probe in greater depth;
- Aim is to get an in-depth understanding of the project objectives, people’s roles and their requirements
- There are two types of interviews
  - **Structured interviews** where the questions are pre-determined and cover certain topics
  - **Unstructured interviews** where there is no pre-defined agenda.
- It is quite predominant and includes:
  - managers to understand objectives
  - staff to understand roles and information needs
  - customers or public as potential users

# Interviewing

- Appropriate situations:
  - most projects
  - at the stage in fact finding when in-depth information is required
- Disadvantages:
  - notes must be written up or tapes transcribed after the interview
  - can be subject to bias
  - can be time-consuming and costly
  - Participant may provide conflicting information in 1-2-1 interviews which then needs to be resolved. To address this issue, **group interview may replace 1-2-1 interviews or can be used as a follow up.**
- Two important factors for effective interviewing
  - The interviewer must be open-minded and willing to listen
  - Interviewee must be given some kind of starting point (e.g. a question) as people find it easier to talk in a defined context.

## Interviews: How to?

- Before the interview
  - Make advance appointment
  - Get permission from authorities if necessary
  - Prepare - make sure your questions are relevant to the interviewee
  - Consider group interviews or workshops if roles are similar to help avoid conflicting information?
- At the start of the interview
  - Arrive on time
  - Introduce the purpose of the interview.
  - Think how you will make records.
- After the interview
  - Transcribe your records / write up your notes asap while the content is fresh in your mind

# Interviews: How to?

- During the interview
  - Help interviewee to stay on the subject
  - Don’t ask very open-ended questions e.g. as ‘Can you tell me what your requirements are?’
  - Summarise answers back to the interviewee
  - Don’t ask biased questions. Gather facts.
  - Be sensitive when using information from other interviews
  - Collect sample documents
  - Do not overrun.
  - At the end, thank the interviewee and make appointment for a further interview if necessary

## Asking the Right Questions

- The following sources provide useful hints on how to interview and what types of questions to ask in interviews
  - Hathaway, A. (2012). Conduct Requirements Interviews to Discover Stakeholder Needs. [online] Available at: http://businessanalysisexperts.com/seven-characteristics-of-a-good-interviewer/ [Accessed 02/10/2023].
  - Sehlhorst, S. (2006). How To Interview When Gathering Requirements. [online] Tyner Blain. Available at: http://tynerblain.com/blog/2006/01/15/how-to-interview-when-gathering-requirements/ [Accessed 02/10/2023].

## Interviewing Practice

- You will practice this during the seminars from Week 3
- Your seminar lead will play the role of the stakeholder.
- Your task would be to come up with the set of the initial questions for Week 3 Seminar
- Check the schedule that will be published

## Questionnaires / Surveys

- Aims to obtain the views of a large number of people in a way that can be analysed statistically
- Appropriate situations:
  - when views of large numbers of people need to be obtained
  - when users are geographically dispersed
  - for systems that will be used by the general public and a profile of the users is required
- It could be conducted via email or using online tools
- It helps to gather opinions as well as facts
- Gathers requirements anonymously

# Questionnaires / Surveys

- Advantages:
  - economical way of gathering information from a large number of people
  - effective way of gathering information from people who are geographically dispersed
  - a well-designed questionnaire can be analysed by computer
- Disadvantages:
  - good questionnaires are difficult to design
  - no automatic way of following up or probing more deeply
  - Questionnaires may suffer from low response rates

# Questionnaires / Surveys

- Steps involved:
  - Determine the objective or the purpose
  - Establish the sample i.e. who and how may participants?
  - Decide on the timing i.e. when & how long to run it?
  - Decide on the tools for distribution (e.g. Google forms).
  - Find ways to reach potential participants (e.g. Social media groups)
  - Produce the cover letter for the survey
  - Design and test the survey (test it before you send it out)
  - Run the survey (may need to send reminders).

## Questionnaires / Surveys

- Cover letter should inform the participants about the following
  - The aims of the purpose of the survey
  - The supervisor’s name and sponsor(s) (if any)
  - How the results will be used
  - That the responses will be confidential
  - Estimated time to complete
  - The deadline for submitting a response
- Cover letter is also used to receive consent from the participants for they agree to participate in view of above
- You will find an example of consent letter on Blackboard

## Questionnaires / Surveys

- Common questions types
  - Binary questions – e.g. yes or no
  - Multiple choice – i.e. chose one or multiple from several option
  - Scaled questions e.g. how often do you … ? Always, usually, sometimes, rarely, never
  - Ranking questions – chose order of preference
  - Open-ended questions – provide own answer

**YES/NO Questions**

| | | | |
|---|---|---|---|
| Do you print reports from the existing system? (Please circle the appropriate answer.) | YES | NO | 10 |

**Multiple Choice Questions**

| | | | |
|---|---|---|---|
| How many new clients do you obtain in a year? (Please tick one box only.) | a) 1–10 | ☐ | 11 |
| | b) 11–20 | ☐ | |
| | c) 21–30 | ☐ | |
| | d) 31 + | ☐ | |

**Scaled Questions**

How satisfied are you with the response time of the stock update?
(Please circle one option.)

| | | | | |
|---|---|---|---|---|
| 1. Very satisfied | 2. Satisfied | 3. Dissatisfied | 4.Very dissatisfied | 12 |

**Open-ended Questions**

What additional reports would you require from the system?

______________________________________________

______________________________________________

______________________________________________

## Questionnaires / Surveys

- Developing questionnaires/survey
  - Learning curve - need to develop the skills
  - Group questions logically e.g. personal details, responsibilities, current setting, preferences, etc
  - Move from general to specifics
  - Be to the point and clear e.g. do not use jargon
  - Do not ask biased questions or lead the answer e.g. Do you also find that the system is x ….?
  - Obtain feedback from your supervisor on the design

## Observation

- Aim is to see what happens, not what people say happens
- Includes:
  - seeing how people carry out processes
  - seeing what happens to documents
  - obtaining quantitative data as baseline for improvements provided by new system
  - following a process through end-to-end
- Appropriate situations:
  - to verify information from other sources
  - when conflicting information from other sources needs to be resolved
  - when a process needs to be understood from start to finish
  - when quantitative data is required

## Observation

- Advantages:
  - first-hand experience of how the current system operates
  - high level of validity of the data can be achieved
  - verifies information from other sources
  - allows the collection of baseline data – how long does it take to do the job?
- Disadvantages:
  - people don’t like being observed and may behave differently which distorts your findings
  - logistical problems for the analyst with staff who work shifts or travel long distances
  - ethical problems with personal data
  - requires training and skill

## Document Sampling

- Aims to find out the information requirements that people have in the system
- Also aims to provide statistical data about volumes of transactions and patterns of activity
- Includes:
  - obtaining copies of empty and completed documents
  - counting numbers of forms filled in and lines on the forms
  - screenshots of existing computer systems
- Appropriate situations:
  - always used to understand information needs
  - where large volumes of data are processed
  - where error rates are high

## Brainstorming

- A group of people can get together in an informal session to come up with new ideas (i.e.with new requirements)
- Relatively short sessions seem to be most productive
- Selection of personnel is critical and a talented facilitator can make difference
- Some rules
  - *Everyone can have their say*
  - *No criticism no matter how crazy the idea is*
  - *No debate – just note it*

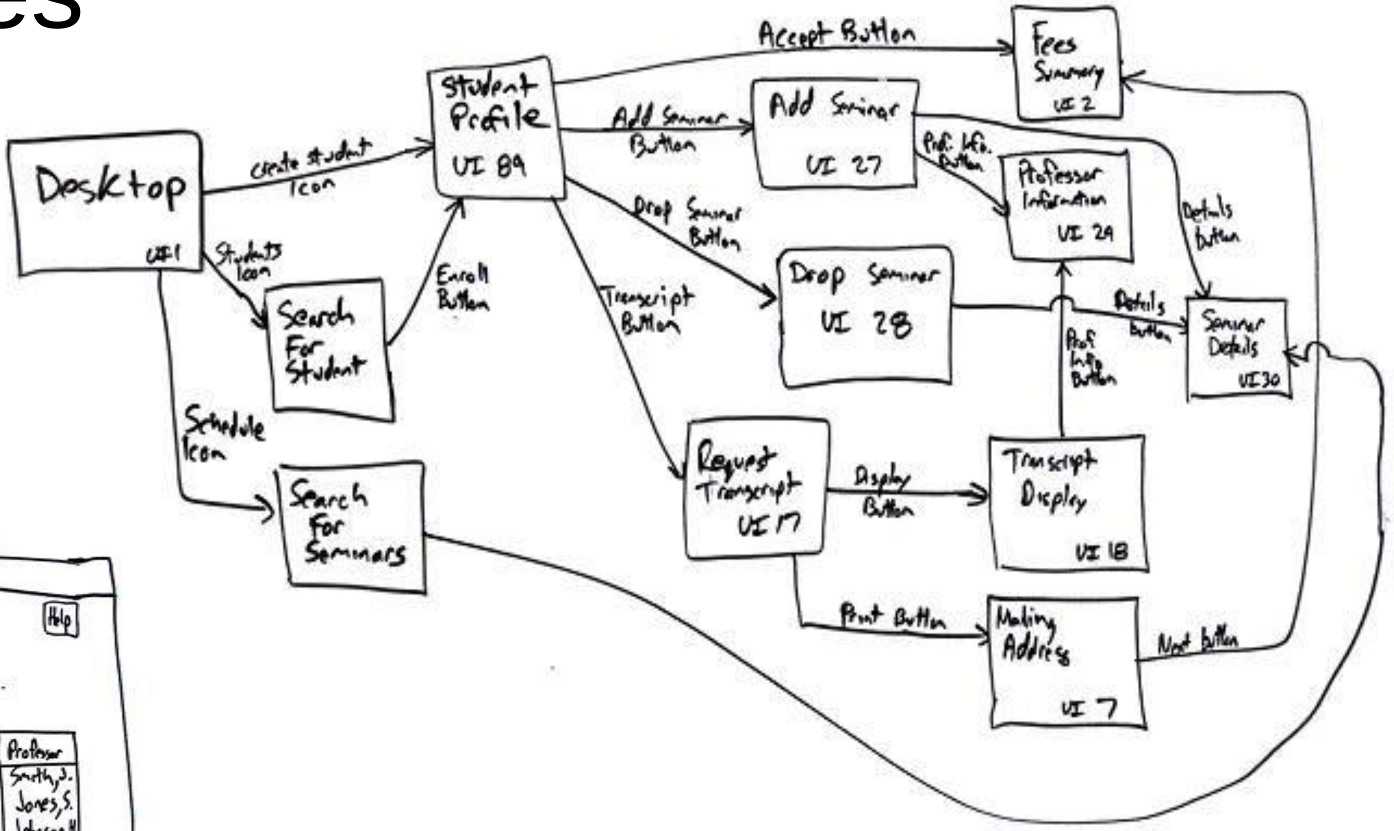
## Workshops

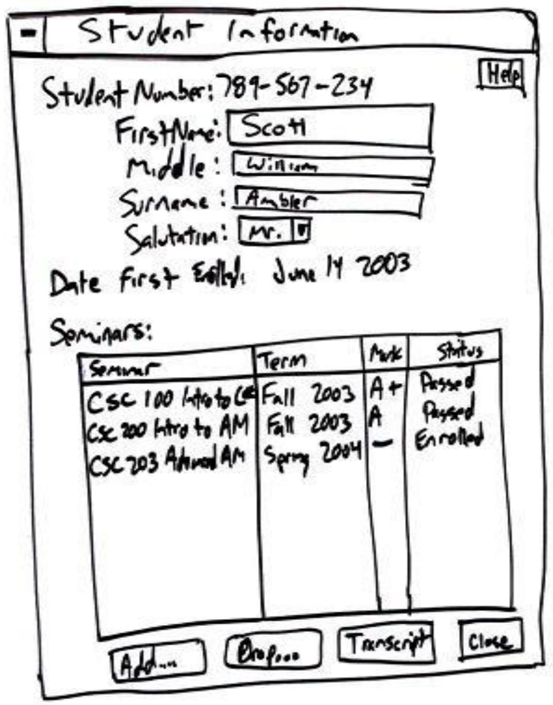
- Also called Joint Application Development workshops
- User involvement is critical to satisfy users
- A group technique that aims to bring  together
  - ***those who have knowledge of business***
  - ***those who have knowledge of system development***
- Structured sessions where the team understands the business processes, project  requirements, agree on decisions etc.
- Participants are *the facilitator, scribe, managers, users and developers*
- They normally take place in a designated period and away from the organisation.

## Prototyping

- Prototyping can be a powerful technique to explore, elicit and validate user requirements
- The users can act as the evaluators of prototypes
- The user reactions to prototypes may contain useful information about their requirements and help
    - *Resolve misunderstandings*
    - *Identify new or missed requirements*

# Low-Fidelity Prototypes





Student Information

Help

Student Number: 789-567-234

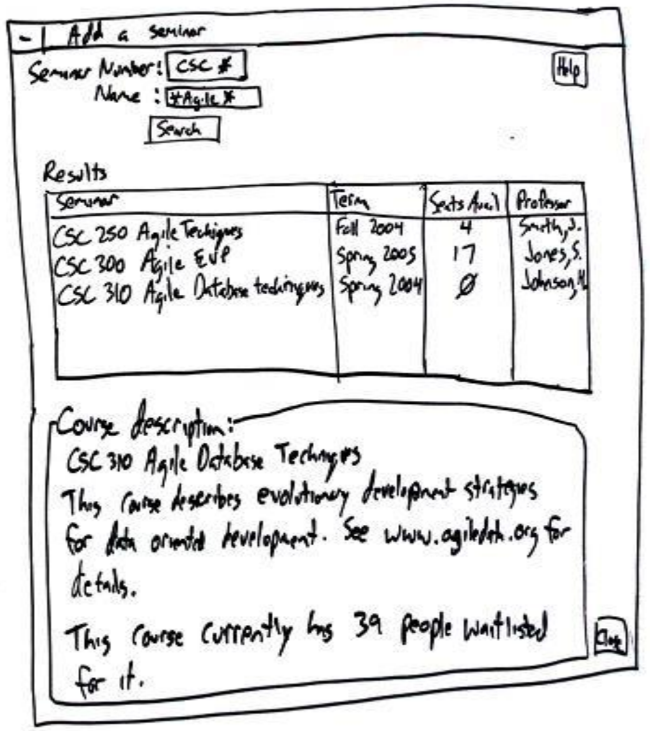
| | |
|---|---|
| FirstName: | Scott |
| Middle: | William |
| Surname: | Ambler |
| Salutation: | Mr. |

Date First Enrolled: June 14 2003

Seminars:

| Seminar | Term | Mark | Status |
|---|---|---|---|
| CSC 100 Intro to CS | Fall 2003 | A+ | Passed |
| CSC 200 Intro to AM | Fall 2003 | A | Passed |
| CSC 203 Advanced AM | Spring 2004 | — | Enrolled |

Add... Drop... Transcript Close



Add a seminar

Help

Seminar Number: CSC #
Name: *Agile*
Search

Results

| Seminar | Term | Seats Avail | Professor |
|---|---|---|---|
| CSC 250 Agile Techniques | Fall 2004 | 4 | Smith, J. |
| CSC 300 Agile EUP | Spring 2005 | 17 | Jones, S. |
| CSC 310 Agile Database techniques | Spring 2004 | Ø | Johnson, H. |

Course description:
CSC 310 Agile Database Techniques
This course describes evolutionary development strategies for data oriented development. See www.agiledata.org for details.
This course currently has 39 people waitlisted for it.

Close

# High-Fidelity Prototypes

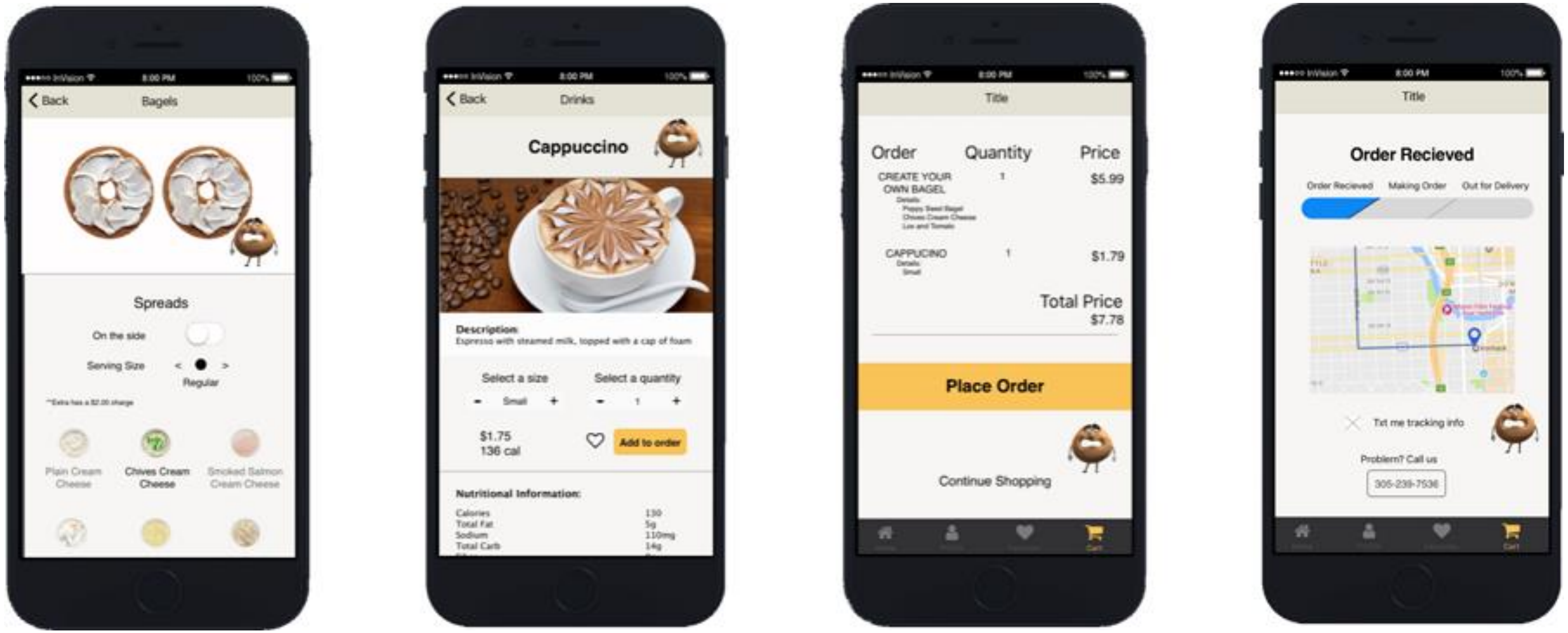


Image source: https://www.mockplus.com/blog/post/high-fidelity-and-low-fidelity

## Selection of Elicitation Techniques

- Selection of elicitation techniques principally depends on previous two considerations i.e.
  - *What information is sought?*
  - *What sources are available?*
- Some techniques are good for gaining a broad perspective, some for filling in details and some for addressing particular areas.
- Certain techniques will require a pre-existing system, access to users etc.
- In practice, economics and time will be a factor.

## Requirements Classification & Organisation

@J.KPNG Modern Analyst
Don't get discouraged. Complex projects are like onions: you have to peel the requirements one layer at time.
Yes, and they also make you cry.

## Levels of Requirements

- Requirement can be high-level abstract statements, detailed descriptions or a formal specification
- We often categorise requirements as
  - *User requirements*
  - *System requirements*
- This categorisation is useful because they communicate information about the system to different types of readers

# User Requirements

- Written for users and customers
- High-level abstract statements
- Describe ***user goals*** or ***tasks*** that the users must be able to perform with the system
- Should describe requirements in such a way that they are understandable by non-technical people
- Should be written in simple ***natural language (NL)***, with simple tables and forms and intuitive diagrams

# User Requirements: Examples

- Different formats exist
  - <type of user> shall/will/should some <task/goal>

    ***Example:***

    Library members shall be able to place reservations on items
  - As a <type of user>, I want <some task/goal> so that <some reason>.

    ***Example:***

    As a student, I want to be able to access my timetable online so that I check where/when my classes are when I need to

## System Requirements

- Detailed descriptions of the system’s functions,  services and operational constraints
- They add detail to user requirements
  - Usually, one user requirement leads to several system requirements
- Define in detail exactly what is to be implemented
- Should be precise, complete and consistent specification of the whole system
- They serve as a basis for the design
- May be part of a contract between client and contractor

## System Requirements

- Can be written
  - In natural language (NL) in text
  - In a more specialised notations e.g. structured language or graphical notations
    - Use case description template
    - Class diagrams
    - Sequence diagrams
    - Activity diagrams
    - .... and similar

# Example – User/System Requirements in NL

1. Library members shall be able to reserve books } **User requirement**

1.1 The system shall allow a member to reserve an item only if no copies of the book is available at that time.

1.2 The system shall allow a member to have up to 10 reservations at any time

1.3 The system shall not allow a member to make a reservation if their account is suspended

1.4 The system shall place reservation on items rather than item copies

1.5 The system shall send an email confirmation containing the reservation number and the book details once the reservation is made.

1.6 ....

**Systems requirements**

# Example - User/System Requirements

**<u>User story</u>**

- As a library member, I want to be able to reserve a book online so that I can borrow it as soon as it is back in the library. (FUNCTIONAL)

**<u>Confirmations</u>**

- Reservation won’t be made if student account is suspended
- The system shall allow a member to have up to 10 reservations at any time
- Reservations will be placed on books titles, not book copies
- Reservation will only be allowed if all copies of a book are out on loan
- The system shall send an email confirmation containing the reservation number and the book details once the reservation is made.
- …..

## Requirements Types

- Requirement conventionally separated as
  - *Functional Requirements*
  - *Non-functional requirements*
- Functional Requirements → Statement of system services (i.e. system capabilities)
- Non-functional requirements → Statement of constraints (e.g. conditions and qualities)

## Functional Requirements

- Statements of services the system should provide
- Describe:
  - *what the system should do*
  - *how the system should behave in a particular situation*
- Include:
  - *processes*
  - *interfaces with users and other systems*
  - *data that the system must hold*

## Functional Requirements

- Can be fairly abstract or detailed depending on whether it expresses a user or system requirement
- Often modelled / documented with diagrams such as
  - Use Case Diagrams
  - Interaction Diagrams
  - Class Diagrams

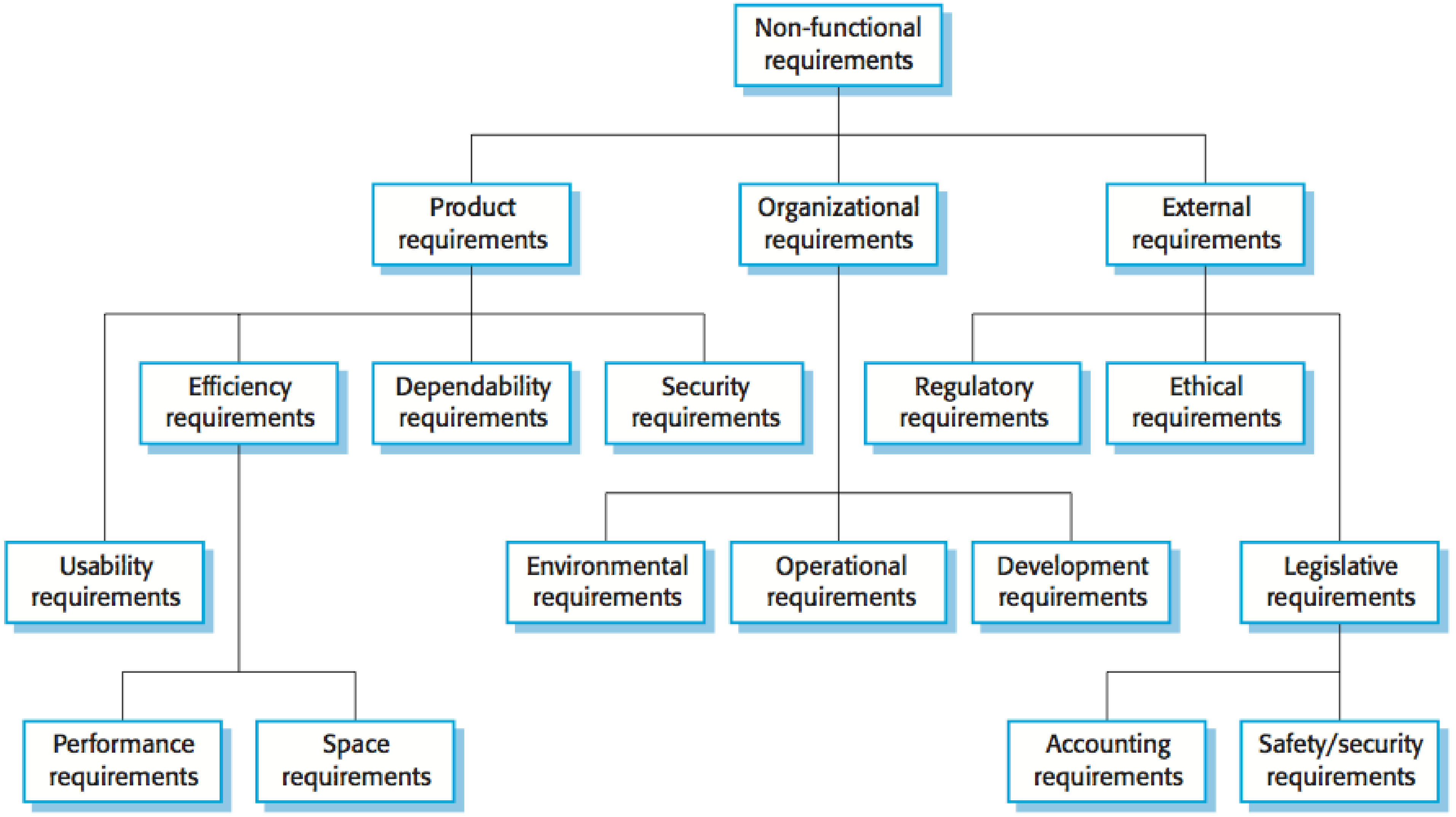
## Functional Requirements: Examples

- Library members shall be able to renew their loans one by one or all at the same time (library system)
- Customers shall have the option to save their credit card details for their future purchases (online retail site)
- Users shall be able to comment on their friends’ photos (a social networking site)
- The system will allow to record a programme while the customer watching another (TV set)
- For a given postcode, the system shall be able to work out the quickest route for a journey (Satnav)
- etc.

## Non-functional Requirements

- Concerned with
  - How the system should behave
  - How well the system should perform
- Include conditions or constrains around
  - Performance – for example response times
  - Capacity – for example, volumes of data
  - Reliability
  - Availability
  - Portability and interoperability
  - Maintainability
  - Usability
  - Security and privacy,
  - … and so on.

# Non-functional Requirements



*Source: (Sommerville, 2016)*

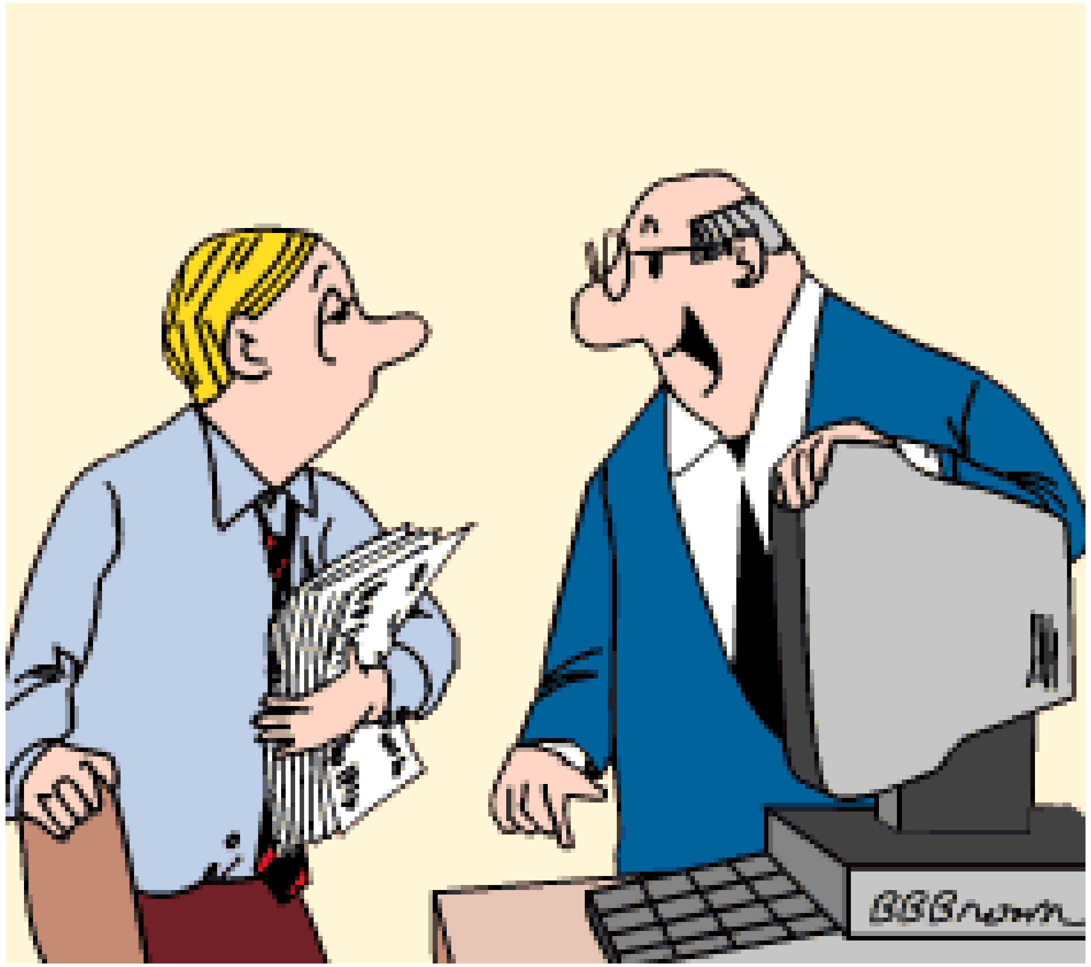
## Non-functional Requirements

- They can be more critical than functional requirements
  - For example, if a system doesn’t meet its reliability requirements it may not be certified as safe for operation
- Some non-functional requirements (NFR) may relate to the development process
- Often documented as a list but may be linked to specific use cases descriptions
- Whenever possible quantify the non-functional requirements so that they are measurable

## Non-functional Requirements: Examples

- The system should be able to handle up to 500 users at the same time without causing any delays to user requests.
- All simulations will complete within 2 hours no matter how complex the problem is (simulation software)
- Sales reports will not be accessible by users, other than store managers (POS system)
- The software shall be compatible with macOS and Windows OS.
- Experienced controllers shall be able to use all the system functions after two hours training. After this training, the average number of errors shall not exceed two per day

## Requirements Prioritisation



"THIS COMPUTER IS EQUIPPED WITH AN AIRBAG IN CASE YOU FALL ASLEEP!"

# Prioritising Requirements

- All requirements are important
- However, all project have constraints such as time, budget, skills, …
- Stakeholders may have conflicting views e.g. not agree on what is essential and what is luxury when too many requirements are identified
- Requirements should be prioritised so that the requirements that delivers the greatest business benefits early

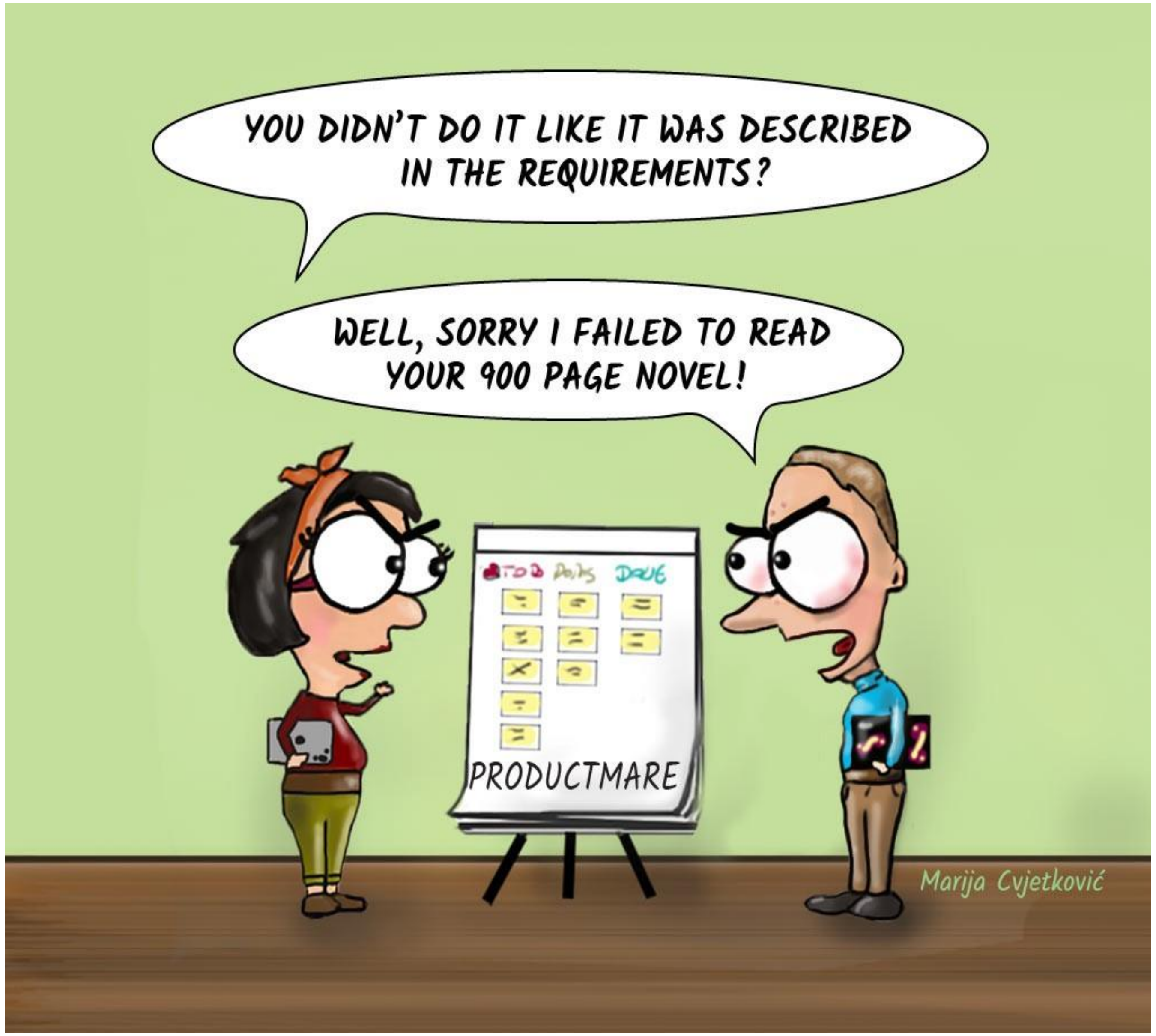
## Prioritising Requirements

- Some software development or project management methods provides guidance & techniques on this
- So, make sure you understands the approach of your method to prioritisation
- For example ‘MoSCoW’ rule is a techniques from DSDM, an Agile Method, to prioritise requirements
  - **M**ust have
  - **S**hould have
  - **C**ould have
  - **W**on’t have (may be in future projects)

## Prioritising Requirements

- Alternatively, requirements can be categorised as:
  - Essential
  - Desirable
  - Out of scope
- Or requirements may be given a weighting e.g. some use this when outsourcing software
  - e.g. 1 to 5 based on their value or importance
  - 1 indicating lowest and 5 indicating the highest value requirements

## Requirements Documentation


YOU DIDN'T DO IT LIKE IT WAS DESCRIBED IN THE REQUIREMENTS?
WELL, SORRY I FAILED TO READ YOUR 900 PAGE NOVEL!
TODO Doing DONE
PRODUCTMARE
Marija Cvjetković

## Requirements Document

- The requirements document is the official statement of what is required of the sodtware/system developers.
- Sometimes called Software Requirements Specification (SRS)
- It should include both the <u>user</u> and a detailed specification of the <u>system</u> requirements.
- It is NOT a design document. As far as possible, it should set WHAT the system should do rather than HOW it should do it

## Requirements Specification

- The chosen development methodology affects the way the requirements are identified and documented.
- Those methodologies based on structured model capture & specify all requirements upfront after the Requirements Analysis Phase.
- Those methodologies based on iterative & incremental model
    - first capture the high level requirements, and
    - develop the details of requirements in each iteration/increment as the project progresses

# Requirements Specification

- Natural Language (NL)
- Structured Language
  - ◻ Standard forms or templates which guides and limit the freedom of the requirements writer hence all requirements are written in a standard way e.g. use case templates
- Graphical Models
  - ◻ Such as Use Case Diagrams, ER Models, Class Diagrams, Activity Diagram, Flowcharts, Sequence Diagrams …
- Other formal methods e.g. Design Description Language or Mathematical Specification

## Problems with NL

- Lack of clarity, ambiguity
  - *Some thing are* only meaningful if you know the exact context.
- Over-flexibility
  - *The same thing may be said in a number of different ways*
- *Lack of modularisation*
  - *NL structures are inadequate to structure system requirements*
- Requirements amalgamation
  - *Several different requirements tend to be expressed together*
- Requirements confusion
  - *Different types of requirements (i.e. Functional and non-functional requirements) tend to be mixed-up*

# Writing Requirements in NL

To minimise misunderstandings

- A standard ***template*** may help. For example,
  - description
  - rationale
  - source
  - reference to the system requirement
- Use language consistently e.g. use '***shall***' for essential and '***should***' for desirable requirements
- Avoid computer jargon as far as possible
- Use text highlighting to identify key parts of requirements

## Modelling Languages

- Use of a standard modelling language addresses the problems with the NL
- Enables unambiguous communication between members of team.
- Help with dealing with complexity
- This can then become a central feature of a quality process.

## Requirements Document - Indicative Content

- Stakeholder Analysis
- Requirements gathering
- User requirements definition
- System architecture
- System requirements specification
- System models
- Appendices
  - Interview transcripts
  - Questionnaire that sent out
  - etc.
  - etc.

# Sources

- I Sommerville, Software Engineering, 10/E, Pearson, 2016 [electronic resource]
- I F. Alexander and L Beus-Dukic, Discovering Requirements: How to Specify Products and Services, Wiley, 2009 [electronic resource]
- K U Wieger and J Beatty, Software Requirements, 3/E, Microsoft Press,  2013 [electronic resource]
- S Bennett, S McRobb and R Farmer, Object Oriented Systems Analysis and Design: Using UML, 3/E, McGraw Hill, 2006.
- Kotonya and Sommerville: Requirements Engineering, Wiley, 1998
- Ian K. Bray: An introduction to Requirements Engineering, Addison Wesley, 2002
- S Lauesen,  *Software Requirements: Styles and Techniques,* Addison Wesley, 2002

## Thank you

## Questions?