EBU6304 – Software Engineering

Requirements

- Topics
  - Getting started → Why do we need requirements and what are they?
  - Theory and Techniques
    - Functional and non-functional requirements
    - Requirements capture techniques
    - Prototyping

What is a requirement?

- A requirement is a feature that your system must have in order to satisfy the customer.
- What the system should do.
- It may range from a high-level abstract statement of a service or a low-level detailed specific system constraint.

Who provides requirements?

1. Customer
2. End user
3. Development team members
4. Management
5. Technology providers

We call these people **STAKEHOLDERS**.

We create software for a reason.
We create software for people.
We need to know what the people want in the software.

These are called REQUIREMENTS.

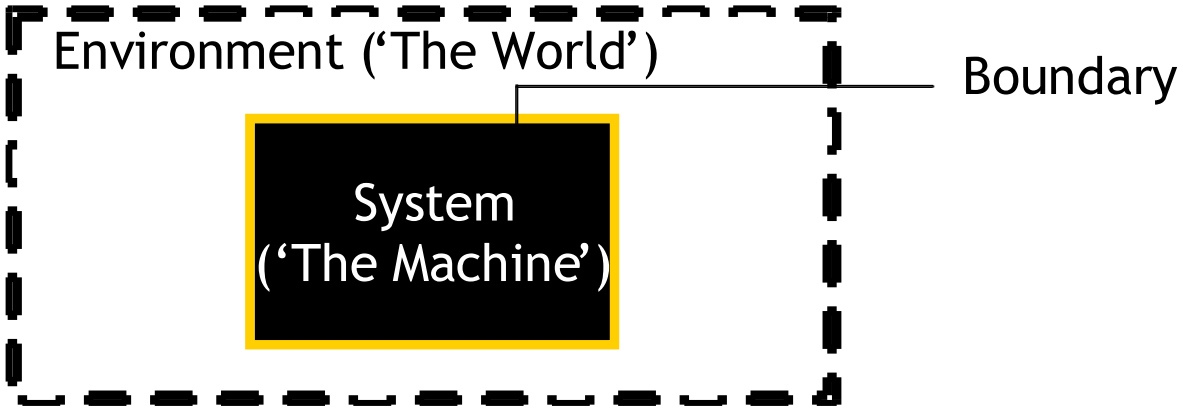
Stakeholder

- Any person or organization who is affected by the system in some way and so who has a legitimate interest
- Stakeholders view the system differently
  - perspective
  - priorities

NOTE: A system may not always have an identifiable stakeholder who sets the requirements. Some software is developed according to an organisation's perception of market demand.

Identify the requirements

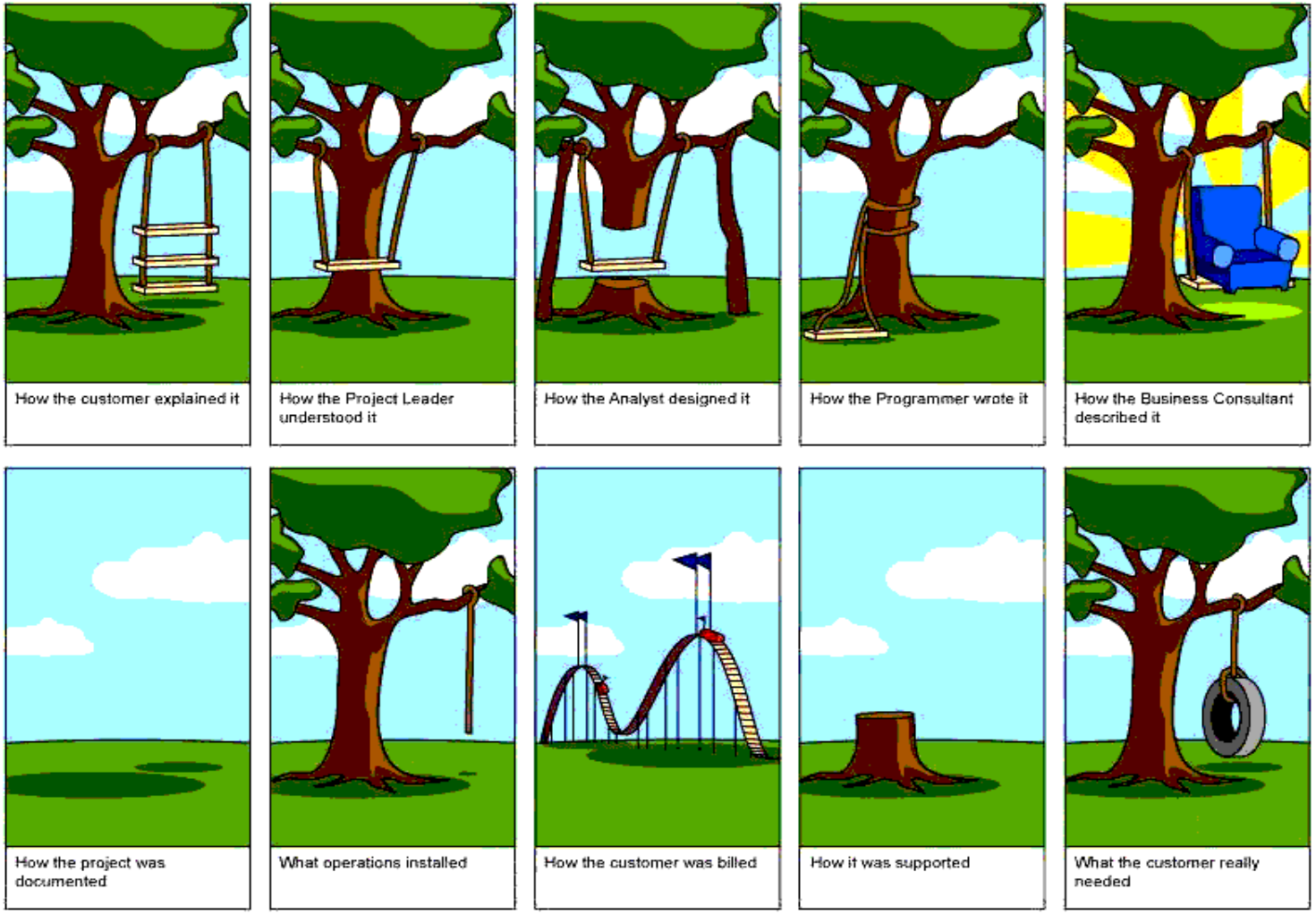
- Talk to the stakeholders
- Ask questions
- Brainstorm
- Look at the problem from many points of view



The problems

- Understanding the requirements is among the most difficult tasks that face a software engineer:
  - Does the customer know what is required?
  - Does the end-user have a good understanding of the features and functions that will provide benefit?
  - What they said = what they mean?
  - Good communication?
  - Even if the answers are all yes: requirements will change, and will continue to change throughout the project!

The classic “The Tire Swing” joke



Why are requirements important?

"The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements, including all the interfaces to people, to machines, & to other software systems. No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later."

Fred Brooks in "The Mythical Man Month"

The Economics of Requirements

Relative cost to fix an error:

Phase in which found	Cost ratio
requirements	1
design	3-6
coding	10
development testing	15-40
acceptance testing	30-70
operation	40-1000

… and these figures are considered conservative!

Requirements Engineering

- “Solution” of those challenges.
- Helps software engineers to better understand the problem they will work to solve.
- Software engineers and other stakeholders all participate in requirement engineering.
- The extra time invested will save time and money in the long term.

Requirement classification

- Functional requirements
  - Services
- Non-functional requirements
  - Constraints: timing constraints, constraints on the development process, standards, etc.

Functional requirements

- Describe what the system should do – Functionality. For example in a Virtual learning environment (VLE) system:

- The lecturer shall be able to set an assignment and its deadline.
- The student shall be able to submit coursework.
- The student shall be able to and view marks.
- The administrator shall be able to enter all the module information.
- The administrator shall be able to create a student list.
- The student shall receive a notification of the announcement.
- The lecturer shall be able to provide feedback comments.

Functional requirements problems

- Imprecision
  - Problems arise when requirements are not precisely stated.
  - Ambiguous requirements may be interpreted in different ways by developers and users.
- Should have …
  - Completeness: include descriptions of all facilities required.
  - Consistency: no conflicts or contradictions in the descriptions of the system facilities.

Non-functional requirements

- Define system properties and constraints
  - Reliability, response time, storage requirements.
  - I/O device capability, system representations, etc.
- Process requirements
  - Quality standard, programming language, development method, etc.
- Non-functional requirements may be more critical than functional requirements.
  - *If these are not met*, the system is useless.

Non-functional examples

- Product requirements (product behaviour)

> The user interface shall be implemented using HTML5.

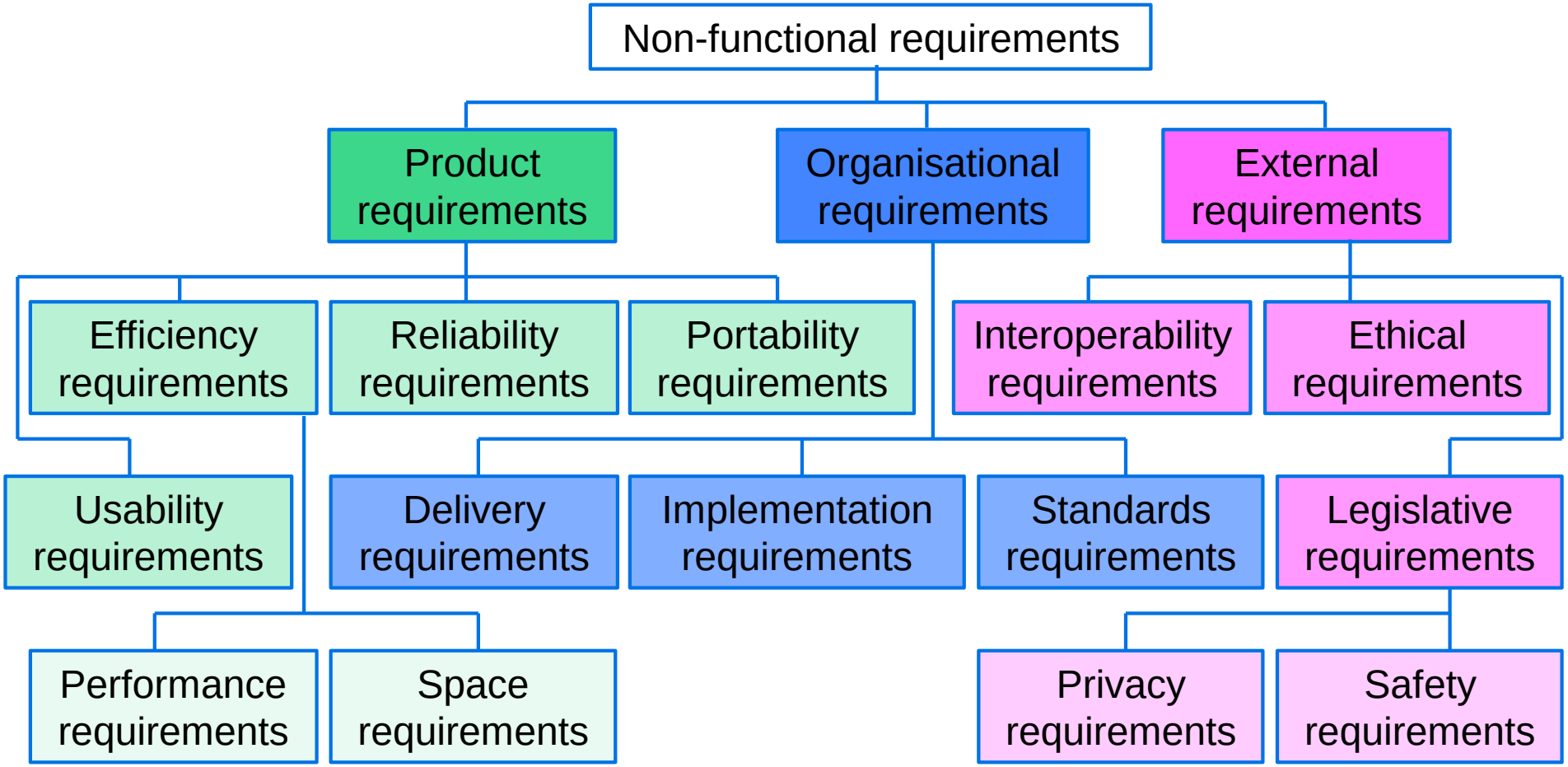
- Organisational requirements (policies and procedures)

> The system development process shall conform to the process defined in the IT policy document.

- External requirements

> The system shall not disclose any personal information about customers, apart from their name and reference number to the operators of the system.

Non-functional requirements



"Software Engineering" by Ian Sommerville, Addison-Wesley

Non-functional requirements problems

- Difficult to verify

The system shall response fast.

The system shall be lightweight.

The system shall be portable.

- Verifiable non-functional requirement
  - Using some measure that can be objectively tested.
  - Quantitative

More Examples of metrics

- System goal

> The system should be easy to use by an experienced administration staff and should be organised in such a way that user errors are minimised.

- Verifiable non-functional requirement

> An experienced administration staff shall be able to use all the system functions after a total of two hours training.

Requirements measures

Property	Measure
Speed	Event response time Transactions/second Auto refresh time
Size	MB/GB
Ease of use	Training time Pages of instructions
Reliability	Rate of availbility Mean time to failure Probability of unavailability Rate of failure occurrence
Robustness	Time to restart after failure Percentage of events causing failure
Portability	Number of target systems Percentage of target dependent statements

Requirements conflicts

- Conflicts between different requirements are common.
- Example: In a digital library system

Requirement A: Item Retrieval

"This system allows the user to retrieve items in any format".

Requirement B: No File Conversion

"File conversion should not be supported".

Requirements conflicts

Which is the most critical requirement? There has to be a trade-off between the requirements.

Options as follows:

1. Support file conversions for all major types and increase the budget for the project.
2. Support file conversions for limited set of formats (e.g. .pdf, .rtf and .doc) and increase the budget for the project.
3. Add the requirement: “A separate version of each item, one for every format required, must be submitted to the system when a new item is added”.”.
4. Only support the retrieval of items in formats that are available.

Choose the best option and this has to be agreed by all stakeholders.

The requirements document

- Software Requirements Specification (SRS)
  - The official statement of what is required of the system developers.
  - It is NOT a design document. As far as possible, it should set out WHAT the system should do, rather than HOW it should do it.

Requirements Capture

- Requirements capture:
  - What is to be built?
  - Different starting points for each project, related to what kind of system is required, the type of organisation that requires the system, the technology etc.

Starting point:
- Determine boundary and interfaces.
- Elaborate functions, features, behaviours.
- Identify data, data transformations.

Fact-finding Techniques

- Fact-finding techniques
  - Background reading
  - Interviewing
  - Observation
  - Document sampling
  - Questionnaires

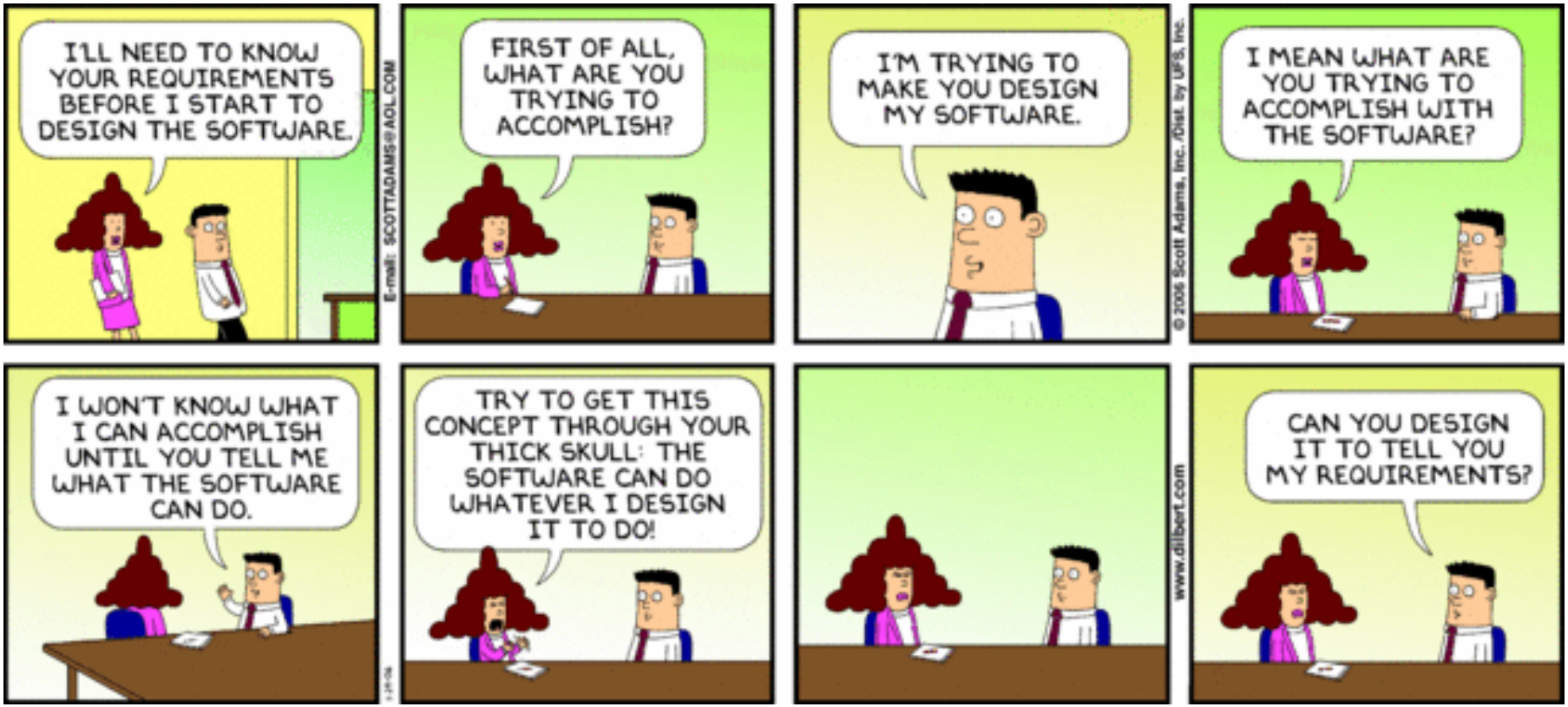
1. Background Reading

- Learn more about the company and department the system is for, using:
  - Company reports
  - Organisation charts
  - Policy manuals
  - Job descriptions
  - Documentation of existing system
- Why do we do this?

2. Interviewing

- Most widely used technique
  - Who?
  - Structured meeting, may be 1-1 or a group.
  - Identify the roles that you want to interview, not the people.
  - Concentrate on identifying the tasks that users complete.
- Start with open-ended questions like
  - "Could you tell me what you do?"
  - "Why", if the developer is an expert
- Move on to more specific questions, to obtain specific information about tasks.
- Direct Contact, Open Mind

Not like this…



http://www.2lucu.com/funny-software/attachment/1125/

3. Observation

- Developer observes the user using the current system
- Provides the developer with a better understanding of the system.
- Often during an interview, the user will concentrate on the everyday tasks and will forget about the “one-off’s”.
- Observation refers to “planned watching of an area under study”
- Findings can be distorted if the user behaves differently.

4. Document or Record Sampling

- A specialist form of observation.
- Different focuses:
  - collect copies of documentation
  - obtain details
  - identify patterns in requirements for the system
- Developer should be aware that the existing forms may not be the best way to model the new system.
- Ideal for capturing non-functional requirements.

5. Questionnaires

- "The aim of questionnaire design is to pose questions where misinterpretation is not possible and there is no bias".
- Economical way of gathering data, ideal for the development of systems to be used by multiple users on multiple sites.
- Results could be analysed by a computer.
- Often suffer from "questionnaire fatigue" and response is therefore low or inaccurate.
- Difficult to write a good questionnaire.

Prototyping

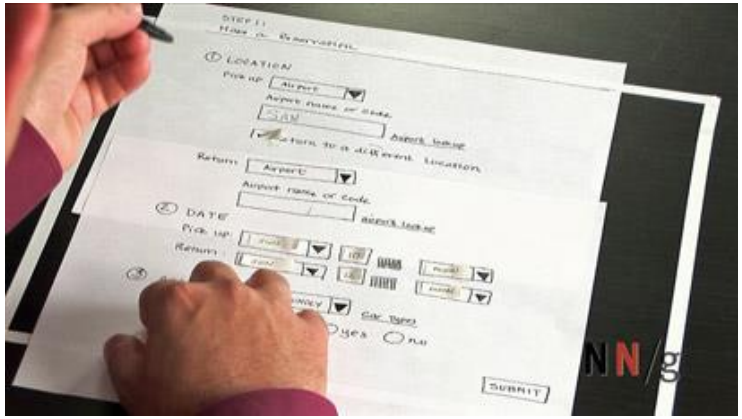
- Physical user-interface design
  - Draw the user interfaces on a paper
    - Get quick feedback
- Logical user-interface design (information)
  - Which user-interface elements are needed?
  - How are these elements related to each other?
  - What should they look like?
  - How should they be manipulated?

Low-fidelity Prototyping

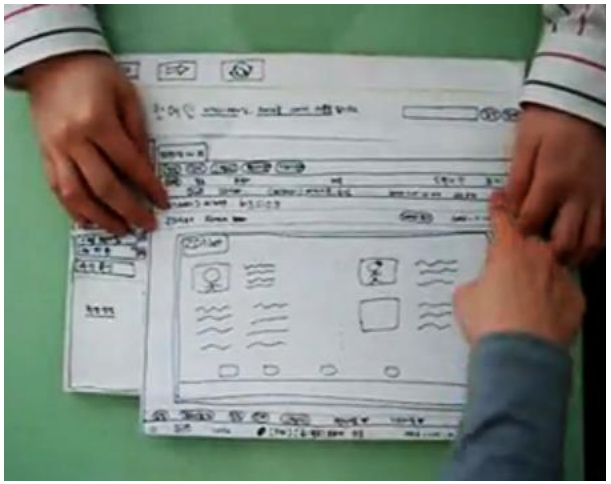
- Paper and sketch

Quick
- No technical learning curve
- Relevant feedback on early concepts
- Provides conceptual direction

Effective
- Low investment of time/resources
- Fail early, fail fast
- Expedites detailed design



http://www.nngroup.com/



Medium-fidelity Prototyping

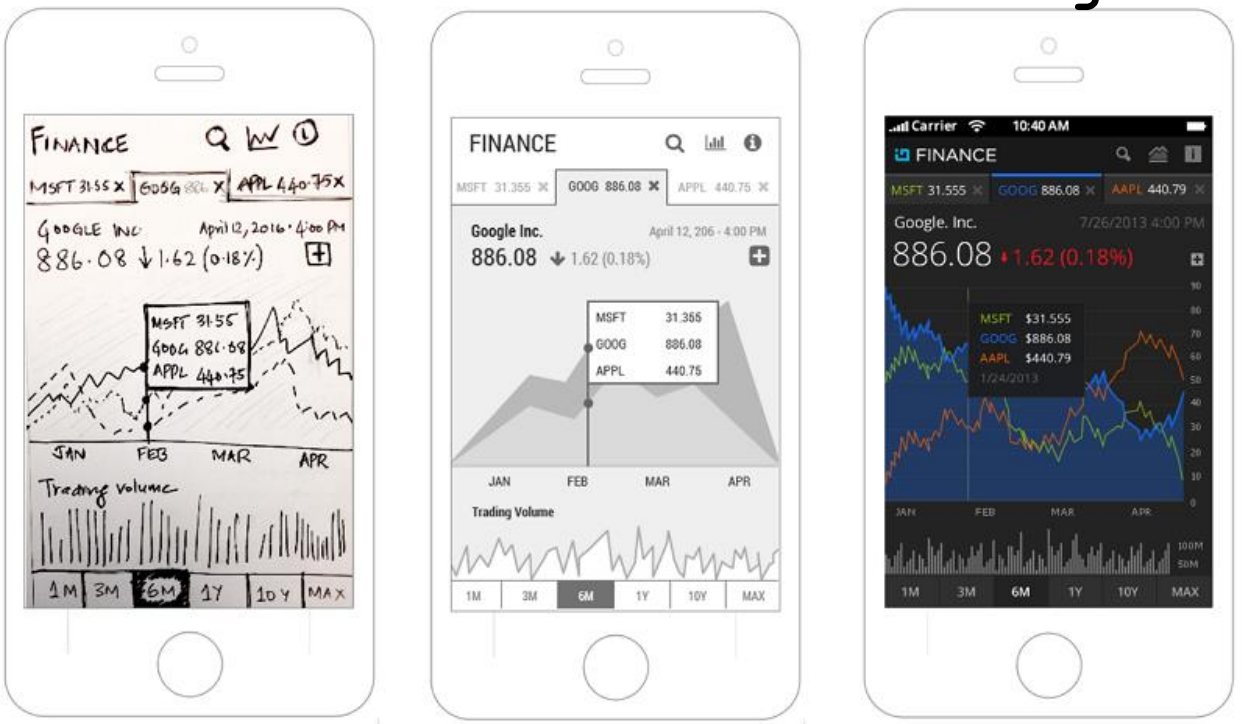
- Limited functionality but clickable areas which presents the interactions and navigation of an application.
- Usually built upon storyboards or user scenarios.
- Tools:
  - Fliplet.com
  - Proto.io
  - Adobe XD
  - And many more

High-fidelity Prototyping

- Computer-based interactive representation of the product in its closest resemblance to the final design in terms of details and functionality.
- Consider user interface (UI) and the user experience (UX).

Examples

Low **Medium** **High**



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

Summary

- Requirements
  - Why do we need requirements and what are requirements?
  - Theory and Techniques
    - Functional and non-functional requirements
    - Requirements capture techniques
    - Prototyping

References

- Chapter 4 – “Software Engineering” textbook by Ian Sommerville
- Chapters 2+3 – “Head First Object Oriented Analysis &Design” textbook by Brett McLaughlin et al
- User stories by Mike Cohn http://www.mountaingoatsoftware.com/agile/
- Introduction to Agile by Sondra Ashmore