SCC141 Week 3: Requirements and requirements engineering

23rd October 2024

Dr Phillip Benachour

With thanks to Dr Emily Winter and Dr Mark Rouncefield for access to previous years' slides

Learning objectives

- To **understand** what requirements are
- To **distinguish** between different types of requirements, especially functional and non-functional
- To **identify** and **understand** the different components of the requirements engineering process
- To **recognize** the pros and cons of various requirement elicitation methods and techniques
- To **evaluate** the challenges surrounding eliciting requirements

Part 1: What are requirements?

What are requirements?

“a requirement is a **statement** about an intended product that specifies **what it should do or how it should perform**”

- Y. Rogers, H. Sharp and J. Preece (2011) *Interaction Design: beyond human-computer interaction*, Wiley.

What are requirements?
Other definitions

- Requirement = ‘simply a statement of **what the system must do or what characteristics it needs to have**’ (A. Dennis, R. M. Roth and B. H. Wixom (2012) *Systems analysis and design,* Wiley)
- **ISO 2007:** ‘a statement that identifies a product or processes operational, functional or design characteristics or constraints, which is **unambiguous, testable or measurable**, and necessary for product or process accessibility’

Why are requirements important?

- Enables ‘time to market with the **right product’** (Hull et al.)
- Central to delivery
- Provide a ‘navigation chart’ for any project (Hull et al.)

Importance of requirements: reasons for project failure

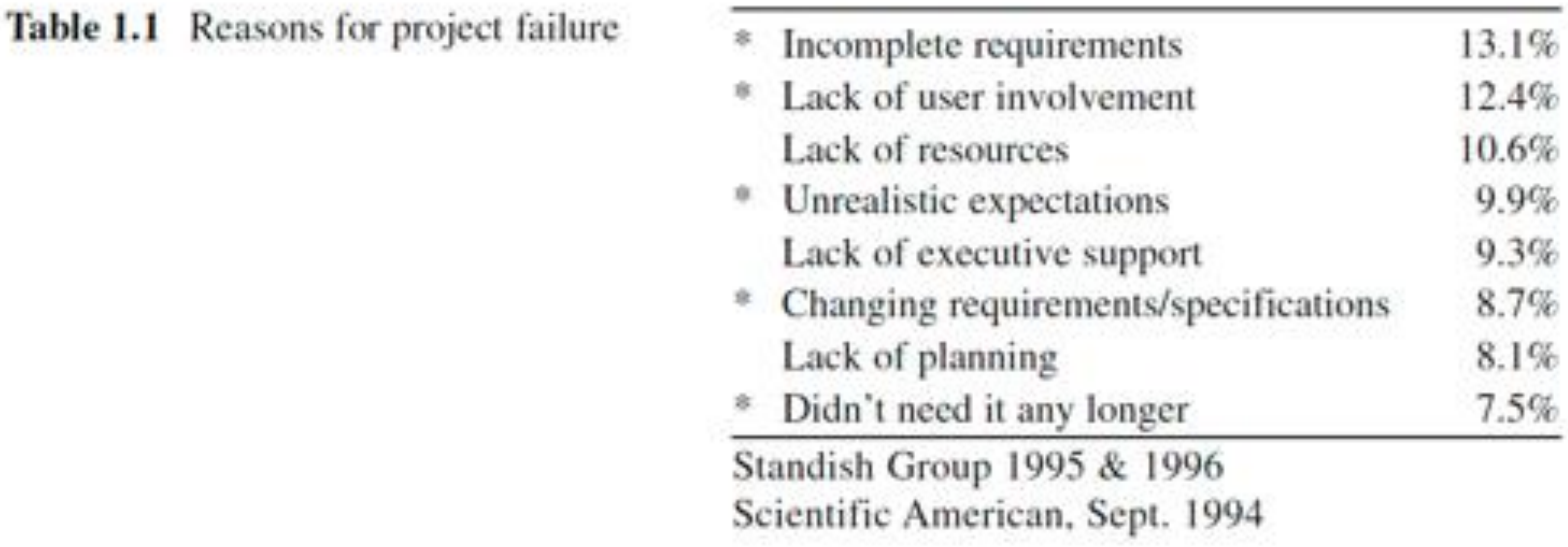


Table 1.1 Reasons for project failure

*	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/specifications	8.7%
	Lack of planning	8.1%
*	Didn't need it any longer	7.5%

Standish Group 1995 & 1996
Scientific American, Sept. 1994

We'll talk more about system and project failures in week 20! ☺

From E. Hull, K. Jackson and Jeremy Dick (2011) *Requirements Engineering*, Springer.

Part 2: Different types of requirements

Different types of requirements

- Business requirements (overall goals of the project)
- User requirements
- Functional requirements (what software should do)
- Non-functional requirements (characteristics the system should have)
- System requirements (how the system should be built)

Functional and non-functional requirements

Functional

- **What the system must do** and how the system will be implemented (functions, data requirements)
- IIBA definition**: ‘the product capabilities or things that a product must do for its users’**

Non-functional

- **Qualities or characteristics that the system must have**; the way the functionality operates
- IIBA definition: **‘the quality attributes, design and implementation constraints, and external interfaces which a product must have’**

Functional and non-functional requirements

Functional

- Processes the system must perform
- Information that the system needs to provide

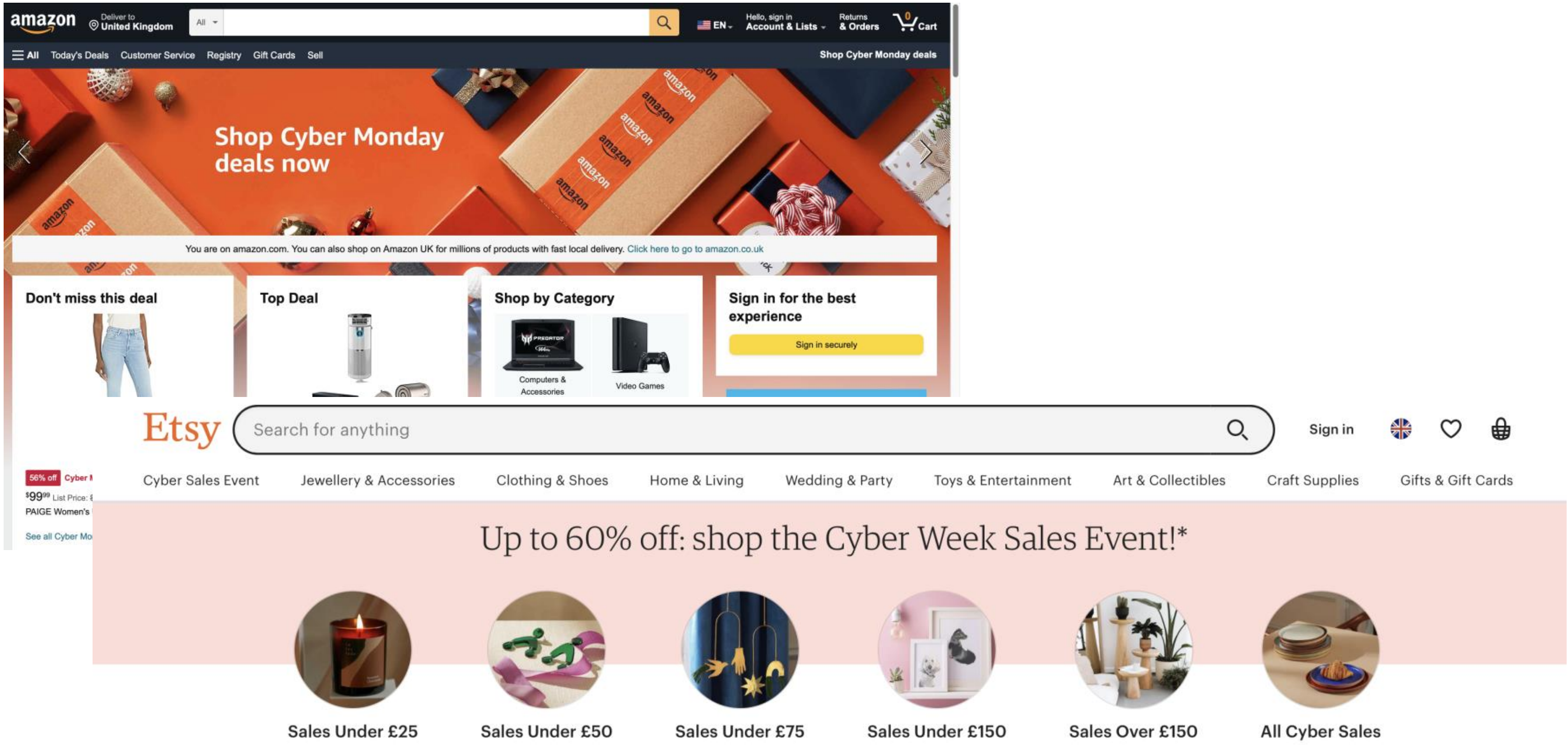
Non-functional

- Usability, performance, maintainability, security, legal compliance, etc.

Non-functional requirements

As important, if not more, than functional requirements (despite the name!)

Example: requirements for online shopping



You are on amazon.com. You can also shop on Amazon UK for millions of products with fast local delivery. Click here to go to amazon.co.uk

Up to 60% off: shop the Cyber Week Sales Event!*

Functional requirements examples

Process:

- **‘The system must allow customers to see their previous 3 years’ order history’**

→Relates to what the system must be able to perform in order to support a user task

Information:

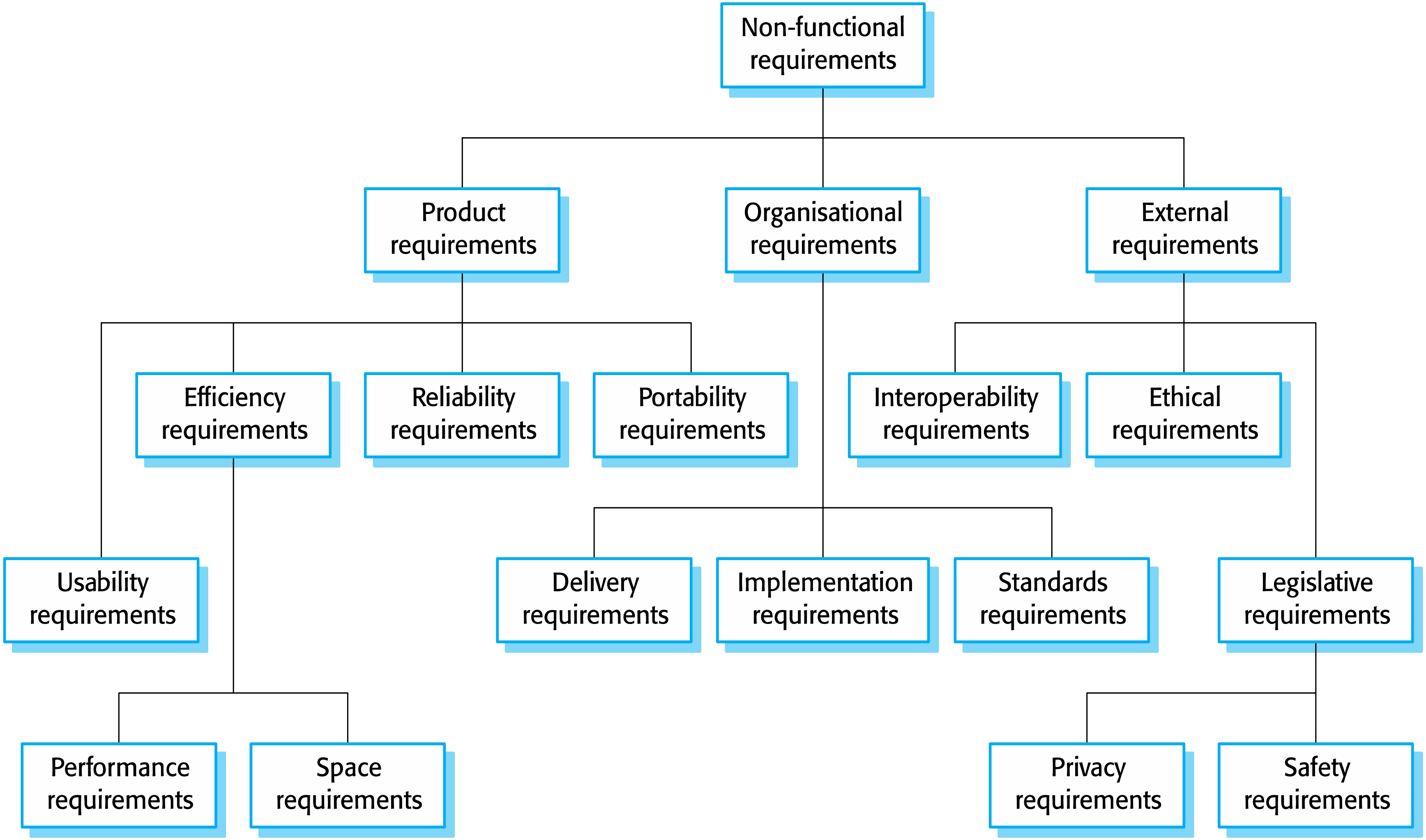
- **‘The system must retain order information for 3 years’**

→Relates to what information the system should contain to enable this process

Types of non-functional requirements

- **Product requirements**
  - Specify that the product must behave in a particular way, e.g., execution speed, reliability, etc.
- **Organisational requirements**
  - Result from organisational policies and procedures, e.g., process standards used, implementation requirements, etc.
- **External requirements**
  - Arise from factors external to the system and its development process, e.g., interoperability requirements, legislative requirements, etc.

Types of non-functional requirements



Non-functional requirements examples

Performance

- ‘It should take no longer than 2 seconds for a user to receive confirmation of their order’

Security

- ‘Users must log-in to a password-protected area to see their order history’

Legal

- ‘Date must be stored in compliance with GDPR’

Another way of categorizing non-functional requirements

- **Quality requirements** – e.g., maintainability, reliability, performance, usability
- **Process requirements** – how the software development process is going to be carried out
- **Constraints** – from business or operational context

Source: Birgit Penzenstadler's YouTube lecture series on requirements engineering: https://www.youtube.com/watch?v=qENBiYaAXNE&list=PLUgFMzuE8lQDeixpbP3s6EyQx8PiNdeQL

What makes a good requirement?

- Clear

- Expressed in natural language without jargon

- Measurable or testable

Part 3: Requirements engineering

Requirements engineering

Barry Boehm:

- Software engineering = **‘designing the thing right’**
- Requirements engineering = **‘designing the right thing’**

- Requirements engineering is about establishing user requirements, i.e., what people want from a computer system
- Idea of getting things right from the start, rather than fixing things later. Very expensive/difficult to fix later!

Requirements engineering – stages/process

- **Requirements elicitation (gathering)**
- Requirements analysis
- Requirements documentation
- Requirements verification and validation

Requirements elicitation

- Articulating and understanding users’ needs, as well as any constraints and any processes that need to be followed
- Understanding the application domain, the problem needing to be solved, organizational needs, specific features and functionality required by stakeholders

Requirements elicitation

May often involve gathering data from (potential) users, which might include:

- Questionnaires
- Interviews
- Techniques like brainstorming
- Observations
- Workshops and focus groups
- Prototyping

Understanding users (more on this next week!)

- ATTITUDES – what people **think** about something
- EMOTIONS – how people **feel** about something
- VALUES – what people consider to be **important**
- BEHAVIOURS – what people **do**
- **These things aren’t always consistent!!**

Questionnaires

- Surveys – asking people for their responses to questions
- Usually quantitative (e.g., rate from 1-5), but can have open-text responses too

Questionnaires

Pros

- Scalable – easy to disseminate to many people, particularly online
- Quantitative insights
- Insight into **attitudes**

Cons

- Often gives little insight into the ‘why’ question
- Not very use for more exploratory, open-ended situations
- Less insight into **emotions** and **behaviours**

Interviews

- 1-2-1 questions
- Usually semi-structured: a set list of questions, but options to ask follow up questions

Interviews

Pros

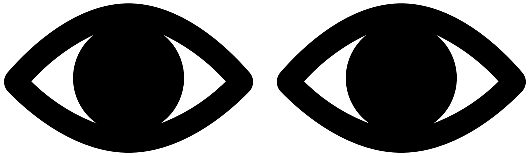
- Flexible
- In-depth and rich insights
- Able to get insight into why people hold certain attitudes

Cons

- Time consuming – not very scalable
- Social desirability factor
- The attitude-behaviour gap

Observations

- Observing what people do – for example, how people use a current system that will be updated
- Well suited to when activities under investigation are hard to articulate in words



Observations

Pros

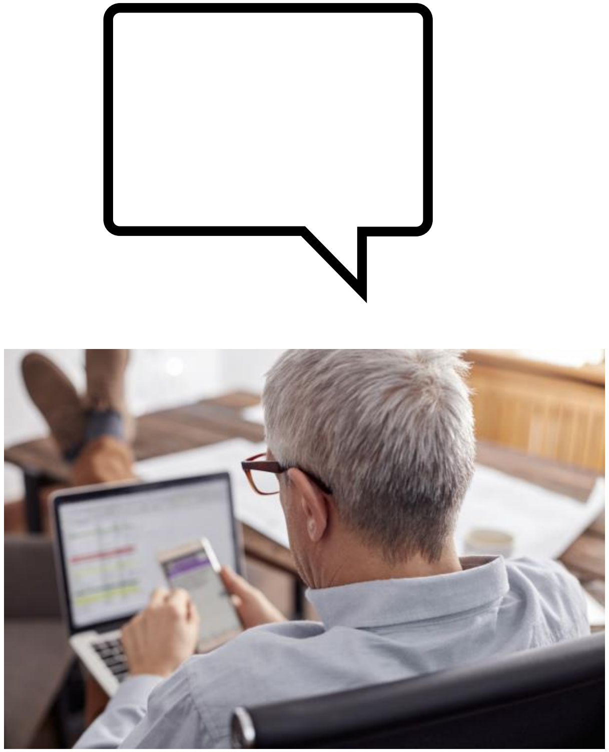
- Insight into people's actual behaviours, practices and activities
- Contextual insights – i.e., seeing activities in the place that they happen

Cons

- Time consuming
- Not easy to do – needs practice!
- Much more challenging in non-work settings (so not great for domestic and personal technologies)
- Do people behave 'naturally' if they feel they're being observed?

Think aloud technique

- Combines observation with some elements of interviewing
- Users talk through what they are doing as they do it
- Aims to gain insight into both the activities and the mental processes behind these activities



Think aloud technique

Pros

- Insight into the thought behind different activities
- Potentially provides more information that just watching someone carry out an activity

Cons

- May be cognitively demanding for participants
- Not all processes have conscious rationale
- Narrating the process may make the process less ‘natural’

Focus groups/workshops

- Similar to interviews, but with more people!
- In workshops, might ask participants to do some kind of activity, etc.

Focus groups/workshops

Pros

- Get insights from multiple people at once
- Get insight into group norms (which may be very influential for how technologies are used)
- Generate rich discussion and often debate

Cons

- Social desirability factor
- Hierarchies and group dynamics
- Can be difficult to manage/facilitate

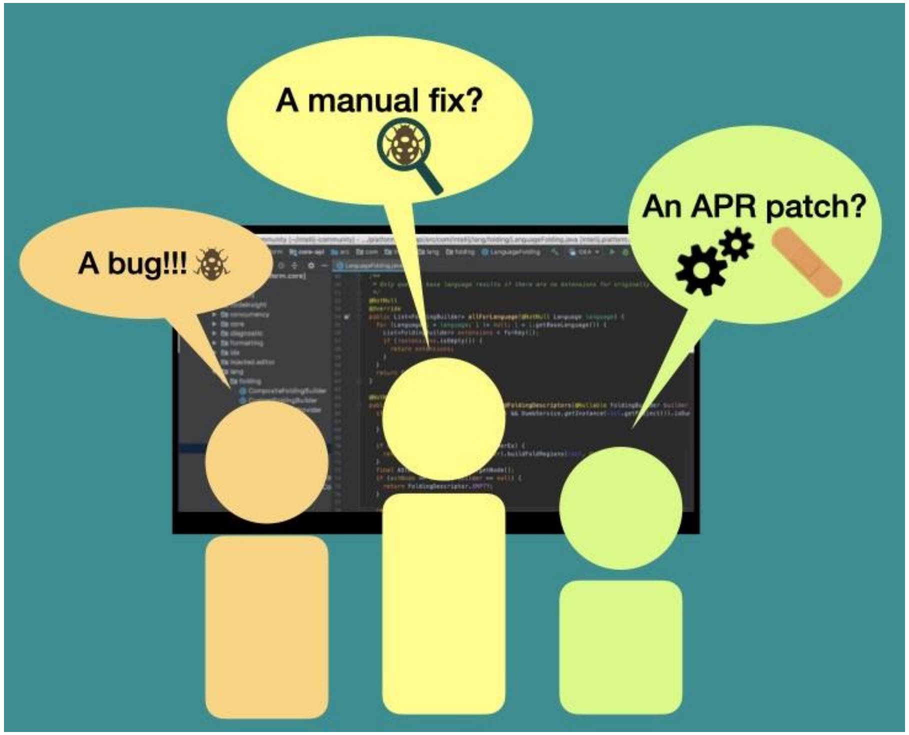
Requirements gathering methods

No perfect method; all have limitations

May often be preferable to use a combination of methods

Eliciting requirements for automatic program repair (APR): case study

- **AIM:** to understand developers' requirements for using an automatic program repair tool
- **METHODS:** questionnaire (~380 developers); focus groups (~20 developers)



Eliciting requirements for automatic program repair (APR): case study

Requirements

- The APR tool should generate patches that are readable, or provide information to explain the patch
- The APR tool should ask developers to review a patch before applying it
- The APR tool should offer developers simple fixes first
- The APR tool should offer developers a series of fixes to choose from

Challenges and limitations

- Are developers always right?!
- Gap between what people say they want and what they do
- Nascent technology

Requirements engineering – stages/process

- Requirements elicitation (gathering)
- **Requirements analysis**
- Requirements documentation
- Requirements verification and validation

Requirements analysis

- Checking over the gathered requirements
- Are there any conflicts/contradictions?
- Is anything missing?

Requirements engineering – stages/process

- Requirements elicitation (gathering)
- Requirements analysis
- **Requirements documentation**
- Requirements verification and validation

Requirements documentation

- Requirements must be expressed and documented in a structured way
- Requirements must be understandable to users
- Requirements usually numbered and may be given importance rankings (e.g., ‘high’, ‘medium’, low’)
- Usually categorized into functional and nonfunctional
- Often accompanied by use cases, process models and data models

Requirements engineering – stages/process

- Requirements elicitation (gathering)
- Requirements analysis
- Requirements documentation
- **Requirements verification and validation**

Requirements verification and validation

"Validation implies getting users to understand the implications of a requirements specification and then agree, i.e. validate, that it accurately reflects their wishes" (Sutcliffe)

- Checking the requirements with users and stakeholders and detecting any differences in understanding and interpretation before the requirements are used to develop system
- **Verifying:** is this correct? Are we building the system right?
- **Validating:** is that what the users want? Are we building the right system?

Lancaster University

Requirements challenges

Challenges of requirements elicitation

It can be hard to find out what users really need. Why?

- Users may not know what they want, or struggle to articulate it
- Users may not be aware of the technical possibilities and constraints

ALSO:

- Difficulties communicating between users and designers – different backgrounds, knowledges, vocabularies and goals
- Tensions between different stakeholder needs

Requirements completeness and consistency

In principle, requirements should be both **complete** and **consistent**.

- **Complete**
  - They should include descriptions of all requirements.
- **Consistent**
  - There should be no conflicts or contradictions in the descriptions of the requirements.

In practice, it is very difficult to produce a complete and consistent requirements document.

Why? Wicked problems

- Most large software systems address **wicked problems,** problems which are very complex and challenging to understand.
- Hard to achieve full understanding
- Therefore, **requirements are normally both incomplete and inconsistent**

Other reasons for inconsistency

- Large software systems aim to improve a current situation → can be difficult to anticipate everything
- Different users have different requirements and priorities → compromise and trade-offs

Devil’s advocate: is it pointless anyway?!

“Some people say, "Give the customers what they want." But that's not my approach. Our job is to figure out what they're going to want before they do. I think Henry Ford once said, "If I'd asked customers what they wanted, they would have told me, 'A faster horse!'" People don't know what they want until you show it to them. That's why I never rely on market research. Our task is to read things that are not yet on the page.”

- Steve Jobs

‘Steve Jobs 1955-2011’ by segagman, licensed under **Attribution 2.0 Generic Creative Commons License**

Next week…

- Users may not always know what they want, but important to consider a range of users anyway
- Always important to acknowledge that not everyone is like you – **you, as the developer or designer, don't necessarily know best!**

Reading and resources

- E. Hull, K. Jackson and Jeremy Dick (2011) *Requirements Engineering*, Springer
- A. Dennis, R. M. Roth and B. H. Wixom (2012) *Systems analysis and design,* Wiley
- Birgit Penzenstadler's YouTube lecture series on requirements engineering: https://www.youtube.com/watch?v=qENBiYaAXNE&list=PLUgFMzuE8lQDeixpbP3s6EyQx8PiNdeQL

Group project distribution of work clarification

Whilst you are free to distribute the work so that, for example, some people are working more on one section that another (for example, leading the writing for a particular section), you need to ensure that **every group member has contributed to every section**. This doesn't necessarily need to be involvement in the writing, but every group member needs to at least do some background research and provide ideas for each of the 2 core sections. **Each section of the report needs to demonstrate that you have worked together on it as a group.**

Lancaster University

Thank you for attending, any questions?