THE UNIVERSITY OF WESTERN AUSTRALIA

Requirements Engineering

Software Requirements and Design
CITS4401
Lecture 3

Key ideas covered last week

- **Stakeholders** are all people and systems that have an interest in a software project
- **Description, rationale and fit criterion** are key elements for specificying a requirement (cf Volere snow cards)
- A **user story** captures a specific business need of a user

Lecture Overview

- What is Requirements Engineering
- Why is it hard ?
- 4 stages of RE
- Functional vs Non-functional requirements

Requirements Engineering

- **Requirements Engineering** is an activity that includes requirements elicitation and analysis
- **Requirements Elicitation** is an activity during which project participants define the purpose of the system.
- **Requirements Analysis** is an activity during which developers ensure that the system requirements are correct, complete, consistent and unambiguous.
- [Bruegge and Dutoit Glossary]

Requirements Example 1

Write a program that will read in a list of 100 positive integers, sort them into ascending order, display the sorted list and display the average of those values

Requirements Example 2

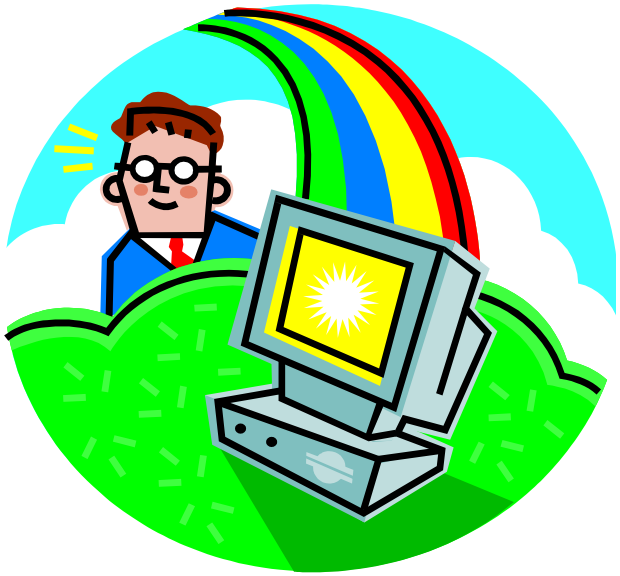
Develop an automated system that will allow us to process orders at least 24 hours sooner, on the average, and will allow us to ship our products to customers at least 3 days sooner than currently

Requirements Example 3

Develop the SW that will allow the Z-676 airliner to land itself, without pilot intervention, at major airports

Requirements Example 4

Develop a new personal productivity product for small computers that will sell at least one million copies at a retail price of at least $200



Why is RE Difficult?

- SW Engineering is a creative, problem solving activity
- Real customers are not sure what they want
- Large SW systems have many different stakeholders with different needs and priorities
- Real developers are not sure how to build it
- Real requirements *creep*

Requirements Tip

It is difficult to capture intent with only “words”

So use some diagrams/pictures

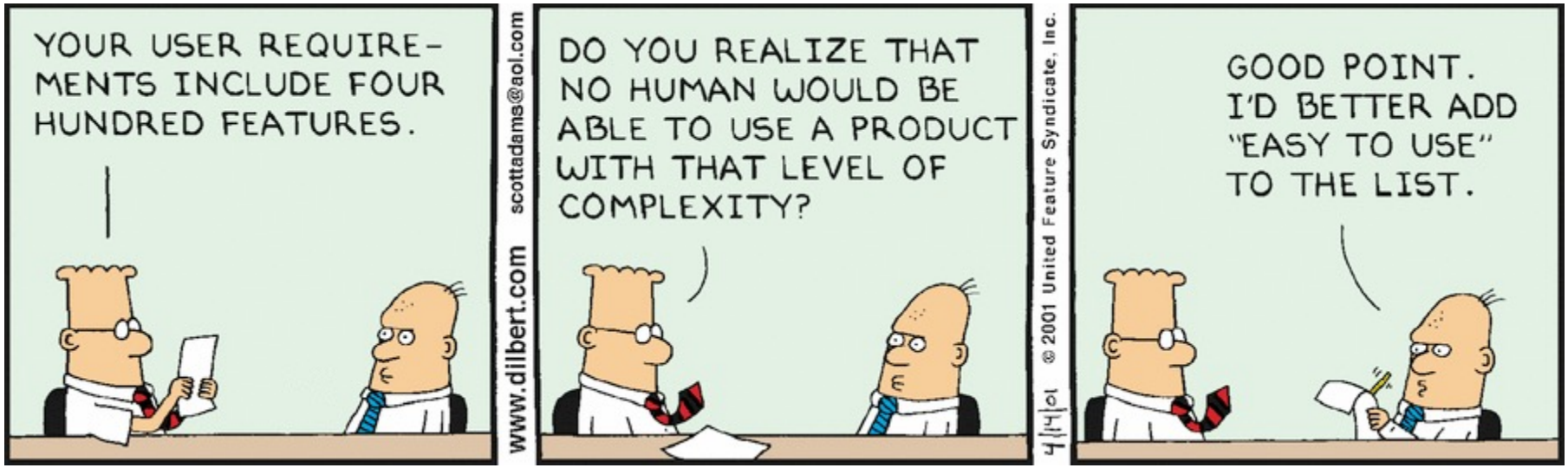
Tip: PowerPoint is a fast & cheap way to mock-up a user interface. Can be 1/10 the time to do mock-ups in code.

Requirements Engineering approach

- a system's users seldom know exactly what they want and cannot articulate all they know
- even if we could state all requirements, there are many details that we can only discover once we are well into implementation
- even if we knew all the details, as humans we can master only complexity to an extent
- even if we could master all the complexity, external forces lead to changes in requirements, some of which may invalidate earlier decisions

[Parnas & Clements 1986]

Complexity ?


YOUR USER REQUIREMENTS INCLUDE FOUR HUNDRED FEATURES.
DO YOU REALIZE THAT NO HUMAN WOULD BE ABLE TO USE A PRODUCT WITH THAT LEVEL OF COMPLEXITY?
GOOD POINT. I'D BETTER ADD "EASY TO USE" TO THE LIST.
www.dilbert.com scottadams@aol.com
© 2001 United Feature Syndicate, Inc.
4/4/01

Why is Requirements Engineering Important?

- Poor requirements capture leads to SW developers solving *the wrong problem* or attempting *an infeasible problem* **(=$$)**
- Misunderstanding the requirements leads to a *chaotic development process* **(=$$)**
- 75% of business and IT executives anticipate their software projects will fail*.
- The U.S. government lost $32 billion to failed IT projects in 2017*

* https://blog.capterra.com/agile-project-management-statistics-for-2018/

Reality is harsh…

Software Project Failures

https://en.wikipedia.org/wiki/List_of_failed_and_overbudget_custom_software_projects

2013	Queensland Health Payroll System	Payroll system	Australia	State government	The Queensland Health Payroll System was launched in 2010 in what could be considered one of the most spectacularly over budget projects in Australian history, coming in at over 200 times the original budget. In spite of promises that the new system would be fully automated, the new system required a considerable amount of manual operation.[13]	$AUD 1.2bn *($6m)*	Outsourced	

2007	2014	e-Borders	Advanced passenger information programme	United Kingdom	UK Border Agency	A series of delays.	over £412m (£742m)	Outsourced	Cancelled
2011	2014	Pust Siebel	Police case management	Sweden	Police	Poor functioning, inefficient in work environments.[9]	SEK 300m ($35m)[10]	Outsourced	Scrapped

The 4 Major Activities of Requirements Engineering

- Elicitation
- Analysis
- Specification
- Validation

The 4 Major Activities of Requirements Engineering

- Elicitation: discover the requirements
- Analysis: ensure the requirements are correct, complete, consistent and unambiguous
- Specification: document the requirements
- Validation: ensure that the system addresses the client's needs

What is a Software Requirement ?

1) A condition or capability needed by a user to solve a problem or achieve an objective.
2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents.
3) A documented representation of a condition or capability as in (1) or (2).

[IEEE Std 610.12-1990]

Source: http://www.computer.org/certification/csdpprep/Glossary.htm

Functional Requirement (Def)

An area of functionality the system must support.

The functional requirements describe the interactions between the actors and the system independent of the realization of the system [Bruegge & Dutoit, Glossary]

Example: The system will display a user's current bank account balance

Non-functional Requirement (Def)

A user-visible constraint on the system.

Non-functional requirements describe user-visible aspects of the system that are not directly related with the functionality of the system. [Bruegge & Dutoit, Glossary]

Example: The system will display user bank account details within 5 seconds

Performance Requirements Note

Often high risk requirements are due to external factors like system load so be careful.

One solution is averaging. eg The system will display user bank account details within 5 seconds when averaged for all such requests over a 24 hour period.

Quality Attributes

- A class of non-functional requirements.
- Examples:
  - usability
  - reliability
  - security
  - Safety

Project Requirements

- **Business Requirements** describe in business terms *what* must be delivered or accomplished to provide value.
- **Product Requirements** describe the system or product which is one of several possible ways to accomplish the business requirements.
- **Process Requirements** describe the processes the developing organization must follow and the constraints that they must obey.

Examples

- A maximum development cost requirement (a Process requirement) may be imposed to help achieve
- A maximum sales price requirement (a Product requirement)
- A requirement for the product to be maintainable (a Product requirement)
- Requirements to follow particular development styles (e.g., OO), style-guides, or a review/inspection process (Process requirements)

Review Questions

Why do you think requirements change so much?

After all, don't people know what they want?

(Some) Review Answers

- People **don't** necessarily know what they want;
- Delivering some SW **may lead to new** requirements;
- SW is **complex** with many stakeholders and many requirements;

- Requirements change so much that it is difficult to predict in advance which software requirements **will persist and which will change**.
- It is equally difficult to predict how **customer priorities** will change as the project proceeds.
- It is often difficult for people to **verbalize** their software needs until they **see** a working prototype and realize that they had forgotten to consider something important. [Pressman 3.5]

Recommended reading

- R. S. Pressman, *Software Engineering: A Practitioner's Approach, 10th* ed., McGraw Hill 2020
  - Chapter 1, covering "**The evoluting Role of Software**", "**The Changing Nature of Software**", "**Software Myths**".
- B. Bruegge and A. H. Dutoit, *Object-Oriented Software Engineering – Using UML, Patterns, and Java,* 3rd ed., Prentice Hall, 2010
  - Section 1.1 "**Introduction: Software Engineering Failures**"
- A. Cockburn, Agile Software Development

Sample Requirements Document Structure

- 1. Introduction
  - 1.1 Purpose of the system
  - 1.2 Scope of the system
  - 1.3 Objectives and success criteria of the project
  - 1.4 Definitions, acronyms, and abbreviations
  - 1.5 References
  - 1.6 Overview
- 2. Current system
- 3. Proposed system
  - 3.1 Overview
  - 3.2 Functional requirements
  - 3.3 Nonfunctional requirements
    - 3.3.1 Usability
    - 3.3.2 Reliability
    - 3.3.3 Performance
    - 3.3.4 Supportability
    - 3.3.5 Implementation
    - 3.3.6 Interface
    - 3.3.7 Packaging
    - 3.3.8 Legal
  - 3.4 System models
    - 3.4.1 Scenarios
    - 3.4.2 Use case model
    - 3.4.3 Object model
    - 3.4.4 Dynamic model
    - 3.4.5 User interface—navigational paths and screen mock-ups
- 4. Glossary