Lecture goals

• Introduce software intensive system
• Overview some software lifecycles and explain RE role
• Define “requirements” and “requirements engineering”
• Explain RE framework
Software-Intensive Systems

• **Software (on its own) is useless**
  – Software is an abstract description of a set of computations
  – Software only becomes useful when run on some hardware
    • we sometimes take the hardware for granted
  – **Software + Hardware = “Computer System”**

• **A Computer System (on its own) is useless**
  – Only useful in the context of some human activity that it can support
    • we sometimes take the human context for granted
  – A new computer system will change human activities in significant ways
  – **Software + Hardware + Human Activities = “Software-Intensive System”**

• ‘**Software**’ makes many things possible
  – It is complex and adaptable
  – It can be rapidly changed on-the-fly
  – It turns general-purpose hardware into a huge variety of useful machines
Software-Intensive Systems

- **Software (on its own) is useless**
  - Software is an abstract description of a set of computations
  - Software only becomes useful when run on some hardware
    - we sometimes take the hardware for granted
  - **Software + Hardware = “Computer System”**

- **A Computer System (on its own) is useless**
  - Only useful in the context of some human activity that it can support
    - we sometimes take the human context for granted
  - A new computer system will change human activities in significant ways
  - **Software + Hardware + Human Activities = “Software-Intensive System”**

- ‘**Software**’ makes many things possible
  - It is complex and adaptable
  - It can be rapidly changed on-the-fly
  - It turns general-purpose hardware into a huge variety of useful machines

Lifecycle of Engineering Project

- **Lifecycle models**
  - Useful for comparing projects in general terms
  - Not enough detail for project planning

- **Examples:**
  - Sequential models: Waterfall, V model
  - Rapid Prototyping
  - Phased Models: Incremental, Evolutionary
  - Iterative Models: Spiral
  - Agile Models: eXtreme Programming
Waterfall Model

- **View of development:**
  - a process of stepwise refinement
  - largely a high level management view

- **Problems:**
  - Static view of requirements - ignores volatility
  - Lack of user involvement once specification is written
  - Unrealistic separation of specification from design
  - Doesn’t accommodate prototyping, reuse, etc.

V-Model

- system requirements
- software requirements
- preliminary design
- detailed design
- code and debug
- component test
- unit test
- software integration
- acceptance test
- system integration

“analyse and design”

“test and integrate”
Prototyping lifecycle

- **Prototyping is used for:**
  - understanding the requirements for the user interface
  - examining feasibility of a proposed design approach
  - exploring system performance issues

- **Problems:**
  - users treat the prototype as the solution
  - a prototype is only a partial specification

Source: Adapted from Dorfman, 1997, p10

Phased Lifecycle Models

- Incremental development (each release adds more functionality)
- Evolutionary development (each version incorporates new requirements)
The Spiral Model

Determine goals, alternatives, constraints

Evaluate alternatives and risks

Plan

Develop and test

Agile Models

Basic Philosophy
- Reduce communication barriers
  - Programmer interacts with customer
- Reduce document-heavy approach
  - Documentation is expensive and of limited use
- Have faith in the people
  - Don’t need fancy process models to tell them what to do!
- Respond to the customer
  - Rather than focusing on the contract

Weaknesses
- Relies on programmer’s memory
  - Code can be hard to maintain
- Relies on oral communication
  - Mis-interpretation possible
- Assumes single customer representative
  - Multiple viewpoints not possible
- Only short term planning
  - No longer term vision

Extreme Programming
Instead of a requirements spec, use:
- User story cards
- On-site customer representative

Pair Programming

Small releases
- E.g. every two or three weeks

Planning game
- Select and estimate user story cards at the beginning of each release

Write test cases before code

The program code is the design doc
- Can also use CRC cards (Class-Responsibility-Collaboration)

Continuous Integration
- Integrate and test several times a day

Source: Adapted from Pfleeger, 1998, p57

Source: Adapted from Nawrocki et al, RE’02
Where is RE in these lifecycles?
What does requirements engineer do?

- Explain the concepts, theories, and best practices associated with requirements engineering
- Elicit, negotiate and document software requirements
- Develop major requirements artefacts and use them during the software development projects
- Apply requirements validation techniques
- Manage software requirements, priorities, and trace them
What are requirements?

- **Domain Properties:**
  - things in the *application domain* that are true whether or not we ever build the proposed system

- **Requirements:**
  - things in the *application domain* that we wish to be made true by delivering the proposed system
    • Many of which will involve phenomena the machine has no access to

- **A Specification:**
  - is a description of the behaviours that the *program* must have in order to meet the *requirements*
    • Can only be written in terms of shared phenomena!

---

**Definition of RE**

Requirements Engineering (RE) is a set of activities concerned with identifying and communicating the purpose of a software-intensive system, and the contexts in which it will be used. Hence, RE acts as the bridge between the real world needs of users, customers, and other constituencies affected by a software system, and the capabilities and opportunities afforded by software-intensive technologies.
Definition of RE

Requirements Engineering (RE) is a set of activities concerned with identifying and communicating the purpose of a software-intensive system, and the contexts in which it will be used. Hence, RE acts as the bridge between the real world needs of users, customers, and other constituencies affected by a software system, and the capabilities and opportunities afforded by software-intensive technologies.

- Not a phase or stage!
- Communication is as important as the analysis
- Quality means fitness-for-purpose. Cannot say anything about quality unless you understand the purpose
- Designers need to know how and where the system will be used
- Requirements are partly about what is needed…
- …and partly about what is possible
- Need to identify all the stakeholders - not just the customer and user

• RE framework
  – System context
  – Core activities
  – Requirements artefacts
  – Validation
  – Management

• **RE framework**
  – System context
  – Core activities
  – Requirements artefacts
  – Validation
  – Management

The part of the system environment relevant for defining, understanding, and interpreting the system requirements.
The part of the system environment relevant for defining, understanding, and interpreting the system requirements
The part of the system environment relevant for defining, understanding, and interpreting the system requirements

- **Subject facet**: objects and events that are relevant for the system, because the system must store or process information about these objects
- **Usage facet**: aspects concerning the usage of the system by people and other systems
- **IT system facet**: aspects concerning the operational or technical environment in which the system is deployed
- **Development facet**: aspects that influence the development of the system imposed by law, or by client and relate to the development process

**RE framework**
- System context
- **Core activities**
  - Requirements artefacts
  - Validation
  - Management

Document important information elicited or developed when performing a core the RE activity

- *i.e.*, documentation, elicitation, negotiation, validation and/or management

Achieve progress in the content dimension by eliciting new requirements as well as detailed information about existing requirements

- Elicit all requirements at the level of detail for the system to be developed
Is there a “Requirements Lifecycle”

Source: Adapted from Pohl, CAISE 1993

Specification
• RE framework
  – System context
  – Core activities
  – Requirements artefacts
  – Validation
  – Management

Intention with regard to objectives, properties, or use of the system

Document sequences of interactions in which the system satisfies some goals or fails to satisfy them
Specify requirements at the required level of detail, the desired properties and features of the system to be developed.

Documenting requirements artefacts

**Requirements artefacts**
- Goals
- Scenarios

**Solution oriented requirements**

**Diagram:**
- Identifier
- Description
- Requirements artefact
- Criticality
- Priority
- Risk
- Goal
- Scenario
- Example of satisfaction
- Solution-oriented requirement
- Contributed to realisation of
- Is realised by
• **RE framework**
  – System context
  – Core activities
  – Requirements artefacts
  – **Validation**
  – Management


---

**System context**

- Subject facet
- Usage facet
- IT system facet
- Development facet

**Validation**

- Consideration of system context
- Execution of RE activities
- Created requirements artefacts

**Validation techniques:**

- Inspection
- Reviews
- Walkthroughs
- Perspective-based reading
- Prototyping
• RE framework
  – System context
  – Core activities
  – Requirements artefacts
  – Validation
  – Management


- Establishing requirements traceability
- Prioritising requirements
- Managing changes of requirements artefacts
Lecture goals

• Introduce software intensive system
• Overview some software lifecycles and explain RE role
• Define “requirements” and “requirements engineering”
• Explain RE framework

Take Home!!!

• RE – not a stage

• RE framework
  – System context
  – Core activities
  – Requirements artefacts
  – Validation
  – Management