What is a System?
Types of System

- **Natural Systems**
  - E.g. ecosystems, weather, water cycle, the human body, bee colony, ...
  - Usually perceived as hard systems

- **Abstract Systems**
  - E.g. set of mathematical equations, computer programs, ...
  - Interesting property: system and description are the same thing

- **Symbol Systems**
  - E.g. languages, sets of icons, street signs, ...
  - Soft because meanings change

- **Designed Systems**
  - E.g. cars, planes, buildings, freeways, telephones, the internet, ...

- **Human Activity Systems**
  - E.g. businesses, organizations, markets, clubs, ...
  - E.g. any designed system when we also include its context of use
    - Similarly for abstract and symbol systems!

- **Information Systems**
  - Special case of designed systems
    - Part of the design includes the representation of the current state of some human activity system
    - E.g. MIS, banking systems, databases, ...

- **Control systems**
  - Special case of designed systems
    - Designed to control some other system (usually another designed system)
    - E.g. thermostats, autopilots, ...

What is **software(-intensive) system**?
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
Where are the challenges?
(when building these systems)

Cost of getting it wrong

• Cost of fixing errors
  – Typical development process:
    requirements analysis ⇒ software design ⇒ programming ⇒
    development testing ⇒ acceptance testing ⇒ operation
  – Errors cost more to fix the longer they are undetected
    • E.g. A requirements error found in testing costs 100 times more than
      a programming error found in testing

• Causes of project failure
  – Survey of US software projects by the Standish group:

<table>
<thead>
<tr>
<th></th>
<th>2004</th>
<th>2008</th>
</tr>
</thead>
<tbody>
<tr>
<td>Successful</td>
<td>16%</td>
<td>26%</td>
</tr>
<tr>
<td>Challenged</td>
<td>53%</td>
<td>46%</td>
</tr>
<tr>
<td>Cancelled</td>
<td>31%</td>
<td>28%</td>
</tr>
</tbody>
</table>

Top 3 success factors:
1) User involvement
2) Executive management support
3) Clear statement of requirements

Top 3 factors leading to failure:
1) Lack of user input
2) Incomplete requirements & specs
3) Changing requirements & specs
Where are the challenges?

Application Domain

Domain properties

Requirements

Machine Domain

Specifications

Programs

Computers

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!
What is engineering?

“Engineering is the development of cost-effective solutions to practical problems, through the application of scientific knowledge”

“...Cost-effective...”
- Consideration of design trade-offs, esp. resource usage
- Minimize negative impacts (e.g. environmental and social cost)

“... Solutions ...”
- Emphasis on building devices

“... Practical problems ...”
- Solving problems that matter to people
- Improving human life in general through technological advance

“... Application of scientific knowledge ...”
- Systematic application of analytical techniques
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.
Some observations about RE

• **RE is not necessarily a sequential process:**
  – Don’t have to write the problem statement before the solution statement
  – (Re-)writing a problem statement can be useful at any stage of development
  – RE activities continue throughout the development process

• **The problem statement will be imperfect**
  – RE models are approximations of the world
  – will contain inaccuracies and inconsistencies
  – will omit some information.
  – analysis should reduce the risk that these will cause serious problems…

• **Perfecting a specification may not be cost-effective**
  – Requirements analysis has a cost
  – For different projects, the cost-benefit balance will be different

• **Problem statement should never be treated as fixed**
  – Change is inevitable, and therefore must be planned for
  – There should be a way of incorporating changes periodically
MTAT.03.306

RE framework

Table of Contents

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

Table of Contents

• 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.
System context

Subject facet
Usage facet
IT system facet
Development facet

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

But design changes the world…

Problem
Situation

change

System

implementation statement

abstract model of world

problem statement
Table of Contents

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

Document important information elicited or developed when performing a core 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.

Achieve agreement among all stakeholders about the requirements.
- has to deal with conflicts about requirements.
Is there a “Requirements Lifecycle”

Source: Adapted from Pohl, CAISE 1993

Table of Contents

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

System context

Core activities
- Documentation
- Elicitation
- Negotiation

Requirements artefacts
- Goals
- Scenarios
- Solution oriented requirements

Validation

Management

Subject facet
Usage facet
IT system facet
Development facet

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.
Table of Contents

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


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

**Validation techniques:**
- Inspection
- Reviews
- Walkthroughs
- Perspective-based reading
- Prototyping
Table of Contents

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

Take Home!!!

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