Outline
1. What to expect in CITS4401
2. SE: what are the problems?
3. Some important concepts
   - Abstraction
   - Product and Process
   - SW Quality
4. Three Process Models

Software Engineering Design
- Is a creative process in which
  - there are few right/wrong answers
  - but nonetheless some designs are (much) better than others.
- Choices must be evaluated and justified.

Requirements Engineering
- A difficulty in SE:
  How to correctly capture the requirements.

“Design and implement a flight booking system using Java RMI. It is assumed that the Company is on the server side, with all the different system classes and functionalities, and the GUI client and interfaces are on the client side.”
- Clear
- Concise
- Doesn’t change
- Feasible
CITS4401 Lectures

- Lectures will present an overview of problems, theory, and techniques for selected topics in SE, with a specific focus on requirements and design.
- Lectures are supported by essential core reading in the textbook (Bruegge & Dutoit), also Sommerville, Pressman and published articles.
- You will need a copy of Bruegge & Dutoit but copies of other material are available in the Science Library closed reserve.

CITS4401 Practicals

- Fortnightly 2 hour workshop group classes for applying SE requirements and design techniques.
- Pre-class preparation not (usually) required. Handouts will be available on the web. Further work after the class is required.
- Class work usually in groups.

Outline

1. What to expect in CITS4401
2. SE: what are the problems?
3. Some important concepts
   - Abstraction
   - Product and Process
   - SW Quality
4. Three Process Models
What is involved in SE? (1)

- **modelling**
  - focus at any one time on only the relevant details. Many different models of the SW system and its domain are used during development

- **creative problem solving**
  - models are used to search for an acceptable solution and the search is driven by experimentation and constrained by time and budget

What is involved in SE? (2)

- **knowledge acquisition**
  - SEs collect data, organize it into information and formalize it into knowledge

- **rationale driven**
  - when acquiring knowledge and making decisions, SEs need to capture the context in which decisions were made and the reasons for those decisions

Why is SE difficult?

The two fundamental problems of software engineering are the management of

  - **complexity** and
  - **change**

Why is there complexity and change?

- **Requirements are complex**
  - The client usually does not know all the functional requirements in advance

- **Requirements may be changing**
  - Technology enablers introduce new possibilities to deal with nonfunctional requirements

- **Frequent changes are difficult to manage**
  - Identifying milestones and cost estimation is difficult

- **There is more than one software system**
  - New system must often be backward compatible with existing system ("legacy system")
  - Phased development: Need to distinguish between the system under development and already released systems
Complexity

- SW systems
  - perform many functions
  - are built to achieve many different, often conflicting objectives
  - comprise many components
  - many components are custom made and complex themselves
  - many participants from different disciplines
  - development process often spans many years
  - difficult to understand completely by any single person

The Boeing-777 uses a mix of proven equipment, many new technologies and some new features. Altogether the digital aircraft contains over $5 \times 10^6$ lines of code.


Complexity

1st year computing projects have approx

- 250 lines of code requiring apx 25 person hours of effort
- so the B-777 code is the size of 20,000 1st year computing projects taking 57 person years

Change

"No matter where you are in the system life cycle, the system will change, and the desire to change it will persist throughout the life cycle."

Bersoff's 1st law of system engineering (quoted by Sommerville)
Change
- Change pervades SW development
- Requirements, design, code, documentation can all change for good reasons
  - Can you think of some examples?
- A change can impact every work product: system model, source code, documentation
  - Which products would be affected by the changes you identified above?

SW is Flexible
Apollo-11 launch
July 1969
Guidance Computer
2K of 16 bit RAM
36K hard-wired core-rope memory
Never failed in flight operations (almost 6 years mean time to failure)

Problems with Flexibility
Kourou, French Guiana, 4th June 1996
Kazakhstan, 5th May 2003

Outline
1. What to expect in SRD
2. SE: what are the problems?
3. Some important concepts
   - Abstraction
   - Product and Process
   - SW Quality
4. Three Process Models
Abstract (essence, summary OED)
- Abstraction enables us to focus only on relevant details of a complex problem
- What makes a good abstraction?
  - It contains all necessary information
  - It omits irrelevant detail
  - Information provided is easy to use
  - Note that the tube map is NOT an accurate map but a schematic one

Product & Process
- Process: A set of activities that is performed towards a specific purpose. Examples of processes include requirements elicitation, analysis, project management, and testing. (B&D)
- Product: a thing produced by a process
  - In order to produce high quality SW we must consider the quality of both product and process
SW Quality

- The production of high quality SW is a goal of almost all developments
- But how can we achieve this goal?
- Constructive methods (before or during)
  - e.g. Refinement of specifications to code; cleanroom; pair programming
- Review methods (after)
  - e.g. Testing; Software and document inspections; Prototyping

Software Project Management

"The deadline for the project is 5pm Tuesday 1st of June 2004. The demonstration will be held on Thursday 3rd of June from 9am-12pm in Room 2.07. Your should submit: Source code and class files for your system. The code has to be well documented. A how to use (or run) file. You will be assessed on ...

- Fixed deadline
- Fixed project time (plus or minus a few late nights)
- Feasible: done 100s of times before

Outline

1. What to expect in SRD
2. SE: what are the problems?
3. Some important concepts
   - Abstraction
   - Product and Process
   - SW Quality
4. Three Process Models

SW Process models

- **Software lifecycle**: All activities and work products necessary for the development of a software system, including
  - Requirements
  - Design
  - Implementation
  - Test
- **Software life cycle model**: An abstraction representing a SW life cycle for the purpose of understanding, monitoring or controlling a SW life cycle
The Waterfall Model of the Software Life Cycle
(from Royce 1970)

- Project Initiation Process
- Concept Exploration Process
- System Allocation Process
- Requirements Process
- Design Process
- Implementation Process
- Verification & Validation Process
- Installation Process
- Operation & Support Process

V-Model: Distinguishes between Development and Verification Activities

- Level of Detail
  - Low
    - Requirements Elicitation
    - Analysis
    - Design
    - Object Design
    - Implementation
    - Unit Testing
    - System Testing
  - High
    - Requirements Elicitation
    - Analysis
    - Design
    - Object Design
    - Implementation
    - Unit Testing
    - System Testing

Problem with V-Model:
Client's Perception is the same as the Developer's Perception

Client's Understanding
Developer's Understanding

Sawtooth Model

- Requirements Elicitation
- Requirements Analysis
- System Design
- Object Design
- Implementation
- System Integration & Test
- Unit Test
- Integration & Test
- Client Acceptance

Sharktooth Model

- Requirements Elicitation
- Requirements Analysis
- System Design
- Object Design
- Implementation
- System Integration & Test
- Integration & Test
- Client Acceptance

Client's Understanding
Manager's Understanding
Developer's Understanding
Comparing Process Models

- Managers love waterfall models
  - Nice milestones
  - No need to look back (linear system), one activity at a time
  - Easy to check progress: 90% coded, 20% tested
- Different stakeholders need different abstractions
  - V-Model, Sawtooth and sharktooth
- Software development is iterative
  - During design: problems with requirements are identified
  - During coding: design and requirement problems are found
  - During testing: coding, design & requirement errors are found
  - Spiral Model

Spiral Model (Boehm) Deals with Iteration

- Identify risks
- Assign priorities to risks
- Develop a series of prototypes for the identified risks starting with the highest risk.
- Use a waterfall model for each prototype development (“cycle”)
- If a risk has successfully been resolved, evaluate the results of the “cycle” and plan the next round
- If a certain risk cannot be resolved, terminate the project immediately

Activities (“Rounds”) in Boehm’s Spiral Model

- Concept of Operations
- Software Requirements
- Software Product Design
- Detailed Design
- Code
- Unit Test
- Integration and Test
- Acceptance Test
- Implementation

For each cycle go through these steps
- Define objectives, alternatives, constraints
- Evaluate alternative, identify and resolve risks
- Develop, verify prototype
- Plan next “cycle”
Determine Objectives, Alternatives and Constraints

Evaluate Alternatives, Identify, resolve risks

Develop & Verify Product

Prepare for Next Activity
Limitations of the Waterfall, V and Spiral Models

- None of these models deal well with frequent change
  - The Waterfall and V models assume that once you are done with a phase, all issues covered in that phase are closed and cannot be reopened
  - The Spiral model can deal with change between phases, but once inside a phase, no change is allowed
- What do you do if change is happening more frequently? ("The only constant is the change")
  - Issue-based models can address this problem – not examinable but see B&D for discussion

Process Maturity

- A software development process is mature if the development activities are well defined and if management has some control over the management of the project
- Process maturity is described with a set of maturity levels and the associated measurements (metrics) to manage the process
- Assumption: With increasing maturity the risk of project failure decreases

Capability Maturity Levels

1. Initial Level
   - Also called ad hoc or chaotic
2. Repeatable Level
   - Minimal documentation to attempt repeated steps
3. Defined Level
   - Process is institutionalized (sanctioned by management)
4. Managed Level
   - Activities are measured and provide feedback for resource allocation (process itself does not change)
5. Optimizing Level
   - Process allows feedback of information to change process itself