COM 6030 Software Analysis and Design

Lecture 1- Introduction

Module homepage http://vista.shef.ac.uk

Introduction

1. Software engineering: definition, products, processes
2. Software engineering – models, management

Outline

- Aims and objectives
- Software engineering definitions
- Software applications
- Software engineering history
- Software myths
- Software failures
- Processes

Reading: Sommerville chapters1,3; Pressman chapters 1,2

References

- Sommerville, Software Engineering, 5th edition, Addison-Wesley, 1996.
- S Bennett, S McRobb R Farmer, Object-Oriented Systems Analysis and Design using UML, McGraw-Hill, 1999.
- P Stevens, R Pooley, Using UML - software engineering with objects and components, Addison Wesley, 2000.
- M Fowler with K Scott, UML Distilled: Applying the Standard Object Modelling Language, Addison-Wesley, 1997.
- R S Pressman, Software Engineering: A Practioner's Approach, 5th edition, McGraw-Hill, 2000 (or the European adaptation by D. Ince).
- T Gilb, Principles of Software Engineering Management, Addison-Wesley, 1988.

Aims and objectives of the course

Aims and Objectives

The aims of this module are:

- to introduce the basic concepts of software development methods, and the need for a professional approach to software system development;
- to describe the activities of software requirements analysis and software design;
- to introduce the diagrammatic notation UML, and its uses in representing analysis and design models of software systems.

http://www.dcs.shef.ac.uk/intranet/teaching/modules/msc/com6030.html

Software Engineering

Software engineering (SE) is the application of systematic, disciplined, quatifiable approach to the development, operation and maintenance of software; that is, the application of engineering to software. – Pressman p18

SE is an engineering discipline concerned with all aspects of software production from early specifications through to maintenace.

SE is the profession that creates and maintains software applications by applying technologies and practices from computer science, project management, engineering, application domain, and other fields.

http://en.wikipedia.org/wiki/Software_engineering

SE vs Computer Science(CS)

SE deals with practical problems

- Complex software products (I)
- Processes (II)
- Methods/Models (III)
- People (IV)

CS is concerned with

- Theories
- Methods

Algorithms, data structures, programs, formal grammars, abstract machines, complexity, numerical methods…

(I) Software is everywhere

Computer software has become a driving force. It is the engine that drives business decision making. It serves as the basis for modern scientific investigation and engineering problem solving. It is a key factor that differentiates modern products and services. It is embedded in systems of all kinds: transportation, medical, telecommunications, military, industrial processes, entertainment, office products… Software is virtually inescapable in a modern world. And as we move into the 21st century, it will become the driver for new advances in everything from elementary education to genetic engineering. – Pressman p.3

Software applications

Potential applications

- System software
- Real-time software
- Business software
- Engineering and scientific software
- Embedded software
- Personal computer software
- Web-based software
- Artificial Intelligence software
- Research software

SE history

- SE introduced first in 1968 – conference about 'software crisis' when the introduction of third-generation computer hardware led more complex software systems than before;
- Early approaches based on informal methodology leading to:
  - delays in software delivery
  - higher costs than initially estimated
  - unreliable, difficult to maintain software
- Need for new methods and techniques to manage the production of complex software
- Even today SE is in a "chronic affliction" state – Pressman p4.

Software myths(SM)

SM – causes of software affliction (apparently reasonable statements, supported by experienced practitioners) propagated from early days of software development:

- Management myths
  - Standards and procedures for building software
  - State of the art software tools and latest computers
  - Add more programmers if behind the schedule
- Customer myths
  - A general description of objectives enough to start coding
  - Requirements may change as the software is flexible
- Practitioner's myths
  - Task accomplished when the program works
  - Quality assessment when the program is running
  - Working program the only project deliverable

Software failures

Complex software systems failures and bugs:

- Therac-25 (1985-1987): six people overexposed during treatments for cancer
- Taurus (1993): the planned automated transaction settlement system for London Stock Exchange cancelled after five years of development
- Ariane 5 (1996): rocket exploded soon after its launch due an error conversion (16-bit floating point into 16-bit integer)
- The Mars Climate Orbiter assumed to be lost by NASA officials (1999): different measurement systems (Imperial and metric)

http://infotech.fanshawec.on.ca/gsantor/Computing/FamousBugs.htm

However…

Important progress:

- Ability to produce more complex software has increased
- New technologies have led to new SE approaches
- A better understanding of the activities involved in software development
- Effective methods to specify, design, implement software have been developed
- New notations and tools have been produced

(II) Processes

A software process (II) consists of a *set of activities* and associated results which lead to the production of a software product (I) Sommerville p43

Fundamental activities:

- Software specification
- Software design and implementation
- Software validation
- Software evolution

Software developed from scratch or by extending and modifying existing systems

Software specifications (Ss)

Ss refers to services requested (*functional aspects*) and constraints (*non-functional component*) – called *requirements engineering*

- Feasibility study
- Requirements elicitation and analysis
- Requirements specification
- Requirements validation

Lead to reports, models, documents

Software design and implementation (I)

Software design process - a set of activities transforming (iteratively) the set of requirements into design products

- Abstract specification of each sub-system
- Component design
- Interface design
- Data structure
- Algorithm design

A set of reports, models (notations), documents is generated

Software design and implementation (II)

Implementation (programming) stage – transforms the design model(s) into code

- Sometimes interleaved with design
- Tools used to (partially) convert into code
- Programming strategies: top-down, bottom-up
- Use of coding standards
- Quality aspects
- Debugging and testing

Software product

Software validation

The validation is the process of checking that “the correct system” was implemented – inspections and reviews

Verification – “building the product right”; formal verification, testing

- Unit testing
- Module testing
- Sub-system testing
- Systems testing
- Acceptance testing

Software evolution

Software evolution process: changes made to a software product after the system development (but not always) - maintenance

- Changes to repair software faults
- Changes to adapt a software system to different operating environment
- Changes regarding system's functionality

Increasingly maintenance is part of system's development (open source, generic frameworks etc)

Summary

- Software engineering covers all aspects of software production
- Historically motivated by a lack of suitable methods to specify and develop complex software systems
- Software engineering includes software products, processes, models and people
- Failures in software production impose adequate approaches and require a limitation of mythical believes
- Software engineering processes cover the whole cycle of developing a software product (specification through to implementation and maintenance)