Overview

In this lecture we work through the topics of the module and identify what I consider to be the most important points to remember.

The lecture notes and slides (on the CS2 Web site) are essential revision material but you will want to read more! I've suggested some basic starting points on the slides which follow.

Software failures

You know about the sorts of pathological problems which can occur on large and small projects:

Documented failures in large projects : *e.g.* Deadline fixation

Misconceptions leading to failure : *e.g.* Timing of benefits

Standards

You know the basics of software standards:

Why they are useful : *e.g.* Repeatability of process

Their legal implications : *e.g.* Fault attribution

The levels at which standards apply : *e.g.* Principles v element standards.

Key organisations producing standards : *e.g.* IEEE, ISO

Examples of standards in key areas : *e.g.* Systems engineering standards

Methodologies

You know the essentials of two contrasting forms of development methodology:

The Unified Process : A highly controlled design method consisting of waterfall iterations within phases.

Extreme Programming : A more adaptive style of design relies on tight design cycles and configuration management.

Reference: Jacobson, Booch and Rumbaugh 1998 *The Unified Software Development Process*, Chapter 1

Economics of quality

You know some of the factors involved in balancing quality against cost:

The means of quality control : *e.g.* inspection v testing

Quality over lifecycles : *e.g.* cumulative quality improvement

Key quality parameters : *e.g.* defect injection v defect reduction

Algorithmic cost models : *e.g.* COCOMO

Reference: Sommerville 1996 *Software Engineering* Chapter 29

Measurement

You know the sorts of things to include in a software measurement plan. In particular, you know:

Some key issues to address : *e.g.* Growth measures.

Means of identifying issues : *e.g.* Risk assessments.

Limitations of measurement : *e.g.* Incremental design means measuring incomplete functions.

Basic estimators : *e.g.* Plot of staff months against number of lines of source code produced.

Reference: Humphrey 1995 *A Discipline for Software Engineering* Chapter 4

Software size

You know several methods for estimating software size:

Consensus methods : *e.g.* Delphi

Population data methods : *e.g.* Fuzzy

Standard component methods : *e.g.* Component estimating

Function based methods : *e.g.* Function point analysis

Reference: Humphrey 1995 *A Discipline for Software Engineering* Chapter 5

Risk reduction patterns

You know how different aspects of projects can create different risks to project success and, for each aspect, you know ways of reducing the risk:

Knowledge inadequacies : *e.g.* Prototype

Teaming : *e.g.* Holistic diversity

Productivity : *e.g.* Gold rush

Ownership : *e.g.* Owner per deliverable

Distractions : *e.g.* Team per task

Training : *e.g.* Day care

Reference:
`members.aol.com/acockburn/riskcata/riskbook.htm`

Verification and validation

You know several techniques for V & V:

Testing : *e.g.* Black/white box testing

Proof of correctness : *e.g.* Temporal logic

You also know how V & V extends beyond individual models:

- Integration testing
- System testing
- Regression testing

Reference: Sommerville 1996 *Software Engineering* Chapters 22, 23 and 24

Goodbye from S.E. ... For Now

You may want to do more software enginering in the next two years. Currently there are two modules for this:

- Software Engineering with Objects and Components 1.
- Software Engineering with Objects and Components 2.

I hope to see you there.