Software Engineering 4 (F22HX3)

Software Testing & Analysis: Summary

Andrew Ireland

Department of Computer Science
School of Mathematical and Computer Sciences
Heriot-Watt University
Edinburgh

Executive Summary

- Testing can never show the absence of defects, but can increase confidence in the quality of a product.
- Testing should shadow all aspects of the software development life-cycle.
- Early feedback on critical design decisions is crucial to success, *e.g.* communication protocols between distributed systems.
- No "silver bullet" – use a range of techniques that provide different perspectives.
- Testing is time-limited so to be cost effective and deliver on time planning and documentation must be taken seriously.

Knowledge

The Testing Life Cycle: testing in relation to the V-model of software development; trace-ability of requirements and change management issues.

Dynamic Analysis: functional testing; structural testing; hybrid testing.

Static Analysis: reviews & inspections (Fagan style); software metrics; flow analysis; slicing; formal verification.

Safety Critical Systems: safety life-cycle; hazard analysis; risk assessment; safety requirements specification; safety case.

Note: the Testing & Analysis component of the module has strong similarities with the content of module 12.2HD3.

Skills

- Cause and effect analysis
- Equivalence partitioning & boundary value analysis
- Code coverage analysis, *e.g.* MCDC testing etc.
- Built-in testing via assertions
- Design-by-Contract: class invariants, pre- & postconditions
- McCabe's Cyclomatic Complexity measure
- Program slicing
- Hazard identification and fault tree style analysis