High Integrity Computing (F23PS2)

High Integrity Software Development (F29HD2)

Summary Lecture

Andrew Ireland

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

Book Work

- Motivations for high integrity computing.
- The role of software standards.
- Safety Integrity Levels (SILs) and how they relate to the development life-cycle.
- Formal methods: levels of rigour; tool support; abstraction; expressiveness; analytical power.
- Code level static analysis and formal verification.

Book Work

- Ada and the SPARK approach to high integrity Ada.
- Data flow and information flow analysis.
- Assertion based program specification and verification: partial and total correctness; verification condition generation and program proof.
- Exception freedom proofs (MSc).
- Proof by mathematical induction: motivations and importance to program proof.
- Analysis of faults and failures (reliability & availability); fault tolerance.

Problem Solving

- Be able to write and interpret relatively small fragments of SPARK, *i.e.* code with core annotations.
- Be able to explain flow analysis and use it to analyze small fragments of SPARK code.
- Be able to generate verification conditions from specifications based upon the small programming language given in lectures.

Problem Solving

- Be able to construct proofs for the verification conditions generated from such specifications.
- Be able to understand and support exception freedom proofs (MSc).
- Be able to prove conjectures that involve recursive definitions via mathematical induction.
- Understand and be able to calculate MTTF, MTTR, MTBF and availability given system (component) FR and RR; Be able to combine reliability measures (series and parallel).