Personal History

◆ Building safety-critical software applications since 1989
◆ Senior member of the team that developed methodology for safety-critical software at Ontario Hydro
◆ Senior member of team that built and verified Darlington Shutdown System One
◆ With colleagues in McSCert, mainly Tom Maibaum and Mark Lawford, working on software certification in nuclear power, medical devices, automotive, financial legacy systems
◆ One of the founders of the Software Certification Consortium

Certification of Software

◆ Systems are certified, not usually just the software
◆ We need to pay special attention to the software in safety-critical systems, since software fails differently from manufactured physical systems
◆ Software engineering is still a relatively young discipline, so we suggest that, to be effective, we need to look at two interrelated topics
   ● How to build safety-critical systems that contain software, so that they can be certified effectively
   ● How to effectively certify safety-critical systems that contain software
Our Goals

◆ “Prove” that
  ● If built according to spec, the application will be safe and effective (validation)
  ● The application is built to spec (verification)
  ● We have high confidence that the application will deliver “safe” behaviour in the face of hardware malfunction (fault tolerant)
  ● The application will continue to be correct and safe over its lifetime (maintainable and evolvable!

Major Principles

◆ Separate safety systems from control systems
  ● Not just for independence, reduction in complexity is crucial to managing (software) safety

◆ Defense-in-depth
  ● This obviously applies to the design/implementation
    ▪ Diversity
    ▪ Avoidance of single points of failure
  ● It also applies to methods for building and certifying the software
    ▪ In the face of uncertainties, we can bolster our confidence by using more than one way to show compliance of any step – review/inspection as well as mathematical analysis/verification, and testing of models and code. With better understanding (in future) we will be able to target coverage better

One Way to Build It
Facilitate Verification
◆ An obvious interaction between building and certifying the software
◆ How can we facilitate verification and certification while building it?
  ◆ Get/Process/Set
  ◆ Provide explicit refinements and verify them in sequence
  ◆ Integrated methods and high-degree of traceability
  ◆ Provide explicit requirements for fault tolerance
  ◆ Record rationale and document defense-in-depth strategies
  ◆ Mathematics and rigour
  ◆ Qualified tools

Certification Methods
◆ An important tripod: People/Process/Product
◆ Certifying that the people involved have the necessary knowledge
  • Acknowledged to be a problem. Need further research to do this effectively, but minimal standards should be enforced now
◆ Certifying the software development process
  • Check compliance to existing and relevant standards
◆ Certifying the product
  • We generally do not do this well at all – testing is not enough

Certify the Product
◆ This must be the main focus of any certification effort
◆ Audits of validation and verification
◆ Assurance/safety cases
Audits
◆ If the certifying agent (CA) has trust in the manufacturer and confidence that an approved process was followed during development
◆ and if the process produces suitable audit points, e.g. inspection/design reviews, verification reports, and so on
  ◆ The CA can audit slices through the system
  ◆ Different slices can be audited effectively by different people

Assurance Cases
◆ The certifying authority may mandate that the manufacturer present an assurance/safety case that documents, in a structured way
  ◆ The claims that are made (often, safety claims)
  ◆ Arguments that support the validity of those claims
  ◆ Backed up by evidence – drawn primarily from the product and process
◆ Opinion: These assurance cases need to be structured in compliance with some consensus standard or they will be too difficult to review effectively

What We Don’t Have Yet
◆ Identified attributes of software related to certification requirements and associated metrics
◆ How various analyses and verification techniques contribute to overall confidence in safety and efficacy
◆ Appropriate templates for assurance cases
◆ How arguments in assurance cases can be objectively and repeatably evaluated
◆ Effective and dependable integrated tool sets that support integrated methods of development (including model driven engineering) and certification
◆ Safe and effective forensic tools