

# Science of Software Design: Architectures for Evolvable, Dependable Systems

*John McDermid, University of York, UK*

## Overview of Problem

Most successful systems evolve. This is true, perhaps especially so, for software systems as their very success leads to new uses, and hence to the demand for new features and capabilities.

Society now depends on computer and software systems for many aspects of business and domestic life. We may require these systems to be secure, highly available, safe (or contribute to system safety), and so on - and we use the term dependability as an "umbrella" embracing all these properties.

We normally demonstrate dependability largely through testing, although increasingly formal analysis is being applied. Generally analysis and testing for the most critical systems consumes around half of the development budget, and productivity is limited to about 1kLoC per person year.

Large, complex systems are hard to evolve without undermining their dependability. Often change is disproportionately costly, because it is not possible to localise the effects of change, so the extent of re-verification is much greater than the extent of change. In the worst case the cost of change is proportional to the size of the system, not the size of the change.

Thus my challenge for the workshop is how to design systems so that they:

- Are initially dependable;
- Can evolve in response to changed requirements preserving dependability;
- It is cost-effective to show that dependability has been preserved.

Also, approaches are needed which work with "small" embedded systems, e.g. vehicle braking systems, and large, complex distributed systems, e.g. GRID-based applications for aircraft engine maintenance management.

## Potential Partial Solutions

I believe that system architectures are pivotal in meeting the above challenge - or at least moving towards a solution.

Requirements are vitally important. Project data shows that, for dependable systems, most of the problems arise from requirements errors. However I do not believe that "fixing requirements" is the best way of addressing the above challenge. For several reasons, the focus needs to be on architecture<sup>1</sup>.

---

<sup>1</sup> In a complete approach to dependability, requirements would need to be addressed too.

First, dependability properties tend to be emergent, and are much more readily modelled and controlled at an architectural level. Second, change impact cannot be assessed purely in requirements terms - to do so needs an understanding of the dependencies created via design commitments - and this can only be seen via the architecture or more detailed design data.

Issues I think need to be addressed to meet the challenge include:

- Expressivity of architectural notation - most are too narrow to address dependability properties, although the Avionics Architecture Description Language (AADL) is an interesting starting point;
- Contracts at module interfaces - development of contracts which control properties necessary to ensure dependability, e.g. timing and failure behaviour as well as functionality;
- Product lines - analysis of likely changes within a line of related products, and design to make it easier to introduce predicted changes;
- Analysis techniques - methods, preferably automated, for assessing dependability properties.

I am prepared to discuss principles and progress on each of these issues.

### **Author's Background**

John A McDermid MA, PhD, FEng, CEng, FBCS, FIEE, FRAeS

John McDermid worked for the UK Ministry of Defence for ten years, then spent 5 years in a software house which is now part of EDS. He was appointed to a chair in Software Engineering at the University of York in 1987 where he built up a new research group in high integrity systems engineering. The group studies a broad range of issues in systems, software and safety engineering, and works closely with the UK aerospace industry. He is Director of the Rolls-Royce funded University Technology Centre in Systems and Software Engineering and the BAE SYSTEMS-funded Dependable Computing System Centre. He also runs the government-funded Defence and Aerospace Research Partnership in High Integrity Real-Time Systems based at York. The main focus of these centres is developing effective techniques for the development and assessment of large-scale safety critical software.

Professor McDermid is author or editor of 6 books, and has published about 280 papers. He was elected a Fellow of the Royal Academy of Engineering<sup>2</sup> in 2002.

He has recently initiated a study with the Royal Academy of Engineering into the difficulties of developing large scale software systems, and would hope to be able to foster a two-way flow between this NSF workshop and the Royal Academy study.

---

<sup>2</sup> The UK equivalent to the US National Academy of Engineering.