

# Focusing Software Education On Engineering

John C. Knight  
*Department of Computer Science*  
*University of Virginia*  
151, Engineer's Way, P.O. Box 400740  
Charlottesville, VA 22904-4740, USA  
[knight@cs.virginia.edu](mailto:knight@cs.virginia.edu)



## 1. The state of software practice

There is ample evidence from a wide variety of applications that the development of large software systems remains a challenge for the majority of practitioners. A study by Lucent [1] found that the defect density in a major telecommunications system was approximately five per 1,000 lines of source code. A study of safety-critical code by German [2] has shown that significant defects remain in software developed to the standard that is required for commercial aircraft applications. Finally, the avionics system for the Lockheed Martin F22 fighter, currently under development, exhibited a mean time to failure of about 1.5 hours earlier this year [3].

## 2. Engineering is the key

Many efforts have been made to remedy the situation, yet the improvements have been localized, often temporary, and usually only marginal. Software can be built properly. Hall and Chapman [4] report that a major security system developed using rigorous methods had an observed defect rate of 0.04 per 1,000 lines. The problem of poor software quality lies not in poor techniques but in the lack of classical engineering approaches in the software industry. The best way to correct this is to rethink and revise the education process that we offer in computer science and computer engineering degrees. We have to teach rigor and we have to instill a culture of engineering into our students. The first course that students see should not be about a language syntax but about the engineering principles involved in software development. Later courses should build on this. The focus of our educational activities must switch to an emphasis

on *engineering* software. The application of rigor should be routine as it is in classical engineering disciplines.

### 3. References

- [1] Yu, W.D., "A Software Fault Prevention Approach in Coding and Root Cause Analysis", Lucent Technologies technical report.
- [2] German, A, "Software Static Code Analysis Lessons Learned", CrossTalk, The Journal of Defense Software Engineering, November 2003.
- [3] "Code Red Emergency", Aviation Week and Space Technology, p. 35, June 9, 2003.
- [4] Hall, A and R. Chapman, "Correctness by Construction: Developing a Commercial Secure System", IEEE Software January/February 2002.