Lecture outline

1. Subject profile: [10-15 mins].

2. Introduction to high integrity systems [25 mins]

3. Example: aircraft with auto take-off and landing systems, controlled by software: who would get on the first iteration of a system with this in it? But *somebody* must do so.

4. How *do* people write software that is able to do this without failing, when applications and entreprise systems fails so routinely? That is the theme of this subject.

5. What do we mean by high-integrity systems?

   (a) Definition: a system: software + hardware + operating procedures.
   (b) Definition: *high integrity system*: systems that must operate with critical levels of security, safety, reliability, or performance.
   (c) Safety-critical, security-critical, mission-critical, and business-critical.
   (d) Types of high integrity system
   (e) Cannot use standard SE processes and methods.

6. Need to not only produce safe systems, but *demonstrate* they are safe (to customers or regulators).

7. Techniques and methods:

   (a) Must demonstrate dependability: availability, reliability, safety, security, confidentiality, integrity, and maintainability.
   (b) Testing is not enough:, a really large set of tests is still usually only a fraction of all possible behaviour: "*Program testing can be used to show the presence of bugs, but never to show their absence!*" — Edsger W. Dijkstra.
   (c) Methods that are otherwise considered too expensive to apply to "standard" software systems become cost-effective in high integrity domains. This is because the cost of failure far outweighs the effort involved required to engineer a dependable product; and because the cost of testing far outweighs more rigorous verification when required.

8. Topics:

   (a) Intro to high integrity systems (today)
   (b) Safety and security engineering
   (c) Modelling and analysis
   (d) HIS design; fault tolerance and detection.
   (e) Assurance: proving program correctness, programming languages for high integrity systems, safe-subset programming languages.

Stanislaus Petrov: lieutenant colonel at Soviet Air Defence Forces.

12:30am, September 26, 1983. When the entire world could come to a Standstill. A nuclear war was going to burst between Soviet Union and United States. Soviet early-warning system was showing that United States had launched five nuclear missiles toward Soviet Union. Procedure dictated that Petrov should press the big red “START” button to launch a retaliatory strike.

Thankfully, nuclear war was averted by a common sense of Petrov. Under immense pressure he reasoned his way not to respond. His logic was; Why would US attack Soviet Union with just five missiles when they had thousands? They would surely want to wipe us out to prevent retaliation.

Later it was identified that early-warning system was showing faulty reports due to an unusual programming fault in the system that failed to detect a rare alignment of sunlight on high-altitude clouds and the satellites. During the post-crisis analysis, many other similar faults were found. A little fault in the system could result in nuclear holocaust. Had he been mistaken, the mistake would have become obvious in minutes: the post’s detection system had a 15-minute advantage over the ground radars