QNX SOFTWARE SYSTEMS

China Technology Innovation Conference 2011

中国技术创新大会

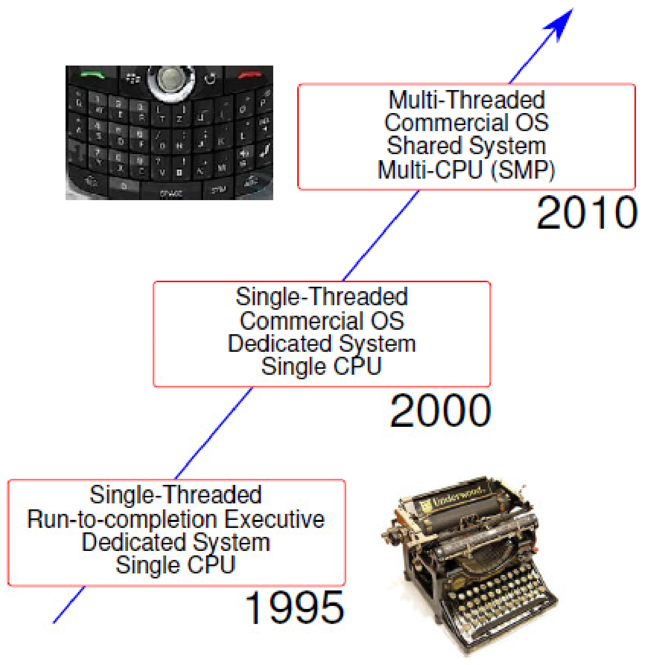
The Changing Nature of Software Failure

Chris Hobbs
Senior Developer, Safe Systems, QNX Software Systems

Dependable Software

- We are relying more-and-more on sophisticated, software-based systems for mission-critical, business-critical and safety-critical applications.

- This software is changing:
  - Companies can no longer write all of the software themselves: they are buying COTS software and making use of Open Source software.
  - The software itself is becoming increasingly complex.



Step Back: What is Software?

Are Field Programmable Gate Arrays (FPGAs) software? Is the firmware in a DSP software? Is the program generated by an automatic program generator software?

Application Note 2: Software and EN 50128 from *Railway Safety* distinguishes between hardware and software:

. . . if a device has few enough internal stored states that it is practical to cover them all in testing, it may be better to regard it as hardware and to show that it meets its safety requirements by analysis of the design and testing of the completed device, including exhaustive testing of all input and state combinations.

(If it’s 100% testable, it’s hardware, not software)

QNX SOFTWARE SYSTEMS

Why is software not 100%testable?

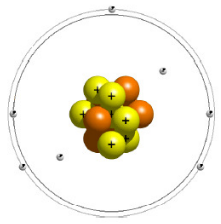
Bohrbugs: the errors only depend on the actual code.

There is no dependency on timing, and applying the

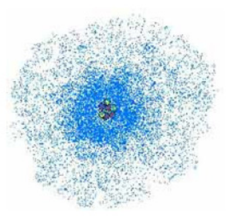
same input values always causes the same error.

Heisenbugs: spirits that appear and disappear—faults found

during testing and reported as “unreproducible”.



Bohrbug:
Nice, solid, easy to find
and correct



Heisenbug:
Difficult to get hold of.
Never sure where it is.

Heisenbugs in Dependable Software?

Example:

- A fault: A multi-threaded program releases shared memory in one thread and then reads it in another.
- An error: On one occasion, the released memory had been given to another thread and overwritten.
- A failure: The thread reads invalid data and crashes.

The fault rarely causes the error and the error normally does not cause a failure. The

failure may occur a long time after the error.

How would testing find this bug?

QNX SOFTWARE SYSTEMS

The Impossibility of 100% Testing

For a complex system, testing has limitations:

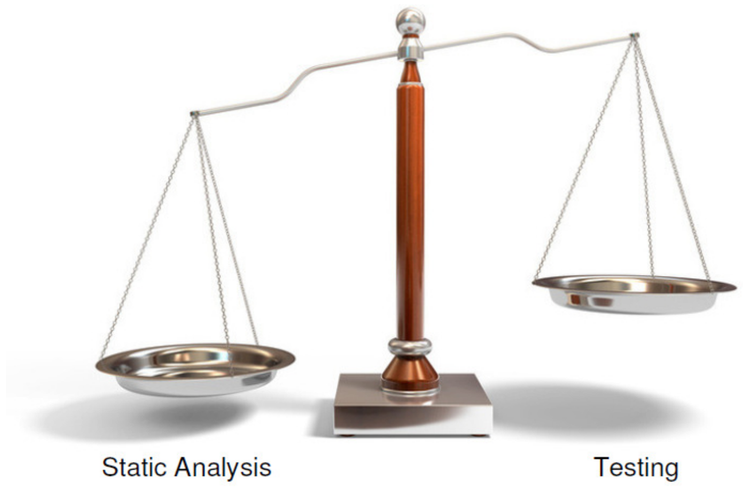
Limited Observability: the tester cannot see how multiple threads interact during a test.

Limited Reproducibility: it is impossible to control the system sufficiently to reproduce the problem.

Testing is a statistical technique: tests can only run a very small number of the possible paths and the results must be handled statistically.

QNX SOFTWARE SYSTEMS

The Balance Changes



The Changing Balance

. . . as testing becomes less useful, static analysis becomes more powerful”.

Evolving Dependable Software

Except for the simplest systems, we must accept that:

- systems will contain Heisenbugs. So use the characteristics of Heisenbugs to guard against them in the design.
- 100% testing is impossible. So treat testing as a statistical activity and back it with static analysis.
- software with different Safety Levels will have to co-exist. So use an operating system that can provide memory and processor time separation.
- systems will include SOUP: Software of Unknown Provenance. So ensure that as much of this software is “clear SOUP”: SOUP with existing certification.

QNX SOFTWARE SYSTEMS

Thank You

QNX SOFTWARE SYSTEMS