Software Testing

Non-functional Testing

and

Choosing a Testing Approach in a Complex World

Section 1 — Testing Non-Functional Properties

Is correctness the thing?

- Testing need not evaluate functional behavior
  - Usability
  - Performance
  - Reliability
  - Maintainability
  - Portability
  - Security…

Exemplar: Performance

- Loosely… will the system work well enough in its intended environment. Will it
  - respond quickly enough? (and fail-to-respond quickly enough?)
  - respond under reasonable load?
  - use a reasonable amount of system resource?
  - …

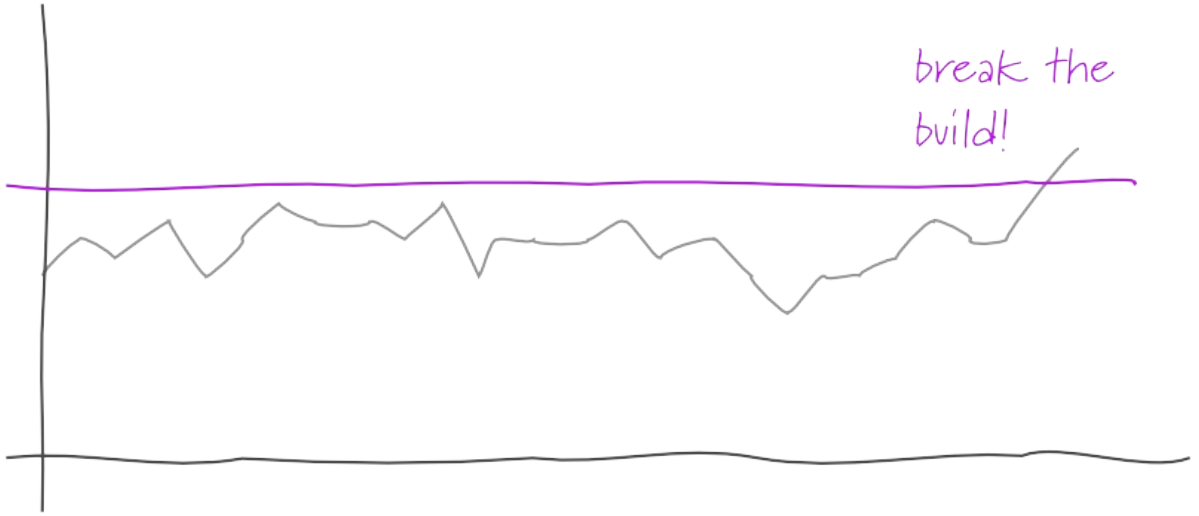
Exemplar: Performance

- **Q:** What is the simplest method you can imagine for evaluating performance?

  (Hint: assume the presence of some automated tests of functional behaviour)

Exemplar: Performance

- “Ratcheting” (Fowler) allows progressive performance enhancements:



http://martinfowler.com/bliki/ThresholdTest.html

Exemplar: Performance

- Simulate realistic loads
  - Aims to satisfy Goldilocks Principle
  - But what is realistic load? Or realistic interaction with system?
  - Can we even know the answer to these questions yet?
  - Example: smart ticketing system – many realistic loads, change over time, won't know what is typical until after deployment and use.

Exemplar: Performance

- Stress testing
  - Aims to quantify the conditions under which a system (or its components) will break
  - Breakages often occur under
    - High load
    - Sustained load
    - Sudden increase in load (“spike”)
    - All of the above

Exemplar: Performance

Stress testing is increasingly popular approach:

"What we needed was a solution that allowed us to use multiple, topographically-close clients to all simultaneously assault our servers with traffic."

http://blog.apps.chicagotribune.com/2010/07/08/bees-with-machine-guns/ (link seems to be dead now)

https://www.wired.com/2010/10/unleash-an-army-of-bees-with-machine-guns-on-your-website/

https://github.com/newsapps/beeswithmachineguns

Exemplar: Performance

- No industry standards that I know of, but…

Response Times: The 3 Important Limits

by Jakob Nielsen on January 1, 1993

Summary: There are 3 main time limits (which are determined by human perceptual abilities) to keep in mind when optimizing web and application performance.

Excerpt from Chapter 5 in my book Usability Engineering, from 1993:

The basic advice regarding response times has been about the same for thirty years [Miller 1968; Card et al. 1991]:

- **0.1 second** is about the limit for having the user feel that the system is **reacting instantaneously**, meaning that no special feedback is necessary except to display the result.
- **1.0 second** is about the limit for the **user's flow of thought** to stay uninterrupted, even though the user will notice the delay. Normally, no special feedback is necessary during delays of more than 0.1 but less than 1.0 second, but the user does lose the feeling of operating directly on the data.
- **10 seconds** is about the limit for **keeping the user's attention** focused on the dialogue. For longer delays, users will want to perform other tasks while waiting for the computer to finish, so they should be given feedback indicating when the computer expects to be done. Feedback during the delay is especially important if the response time is likely to be highly variable, since users will then not know what to expect.

Key Points

- Testing isn’t only for evaluating correctness
- Non-functional qualities have their own evaluation methods, standards and tools
- Familiarity with non-functional evaluation can inform functional evaluation

- Don’t re-invent wheels.
- But do steal them.

- Any questions?

Section 2 — Making Decisions About Testing

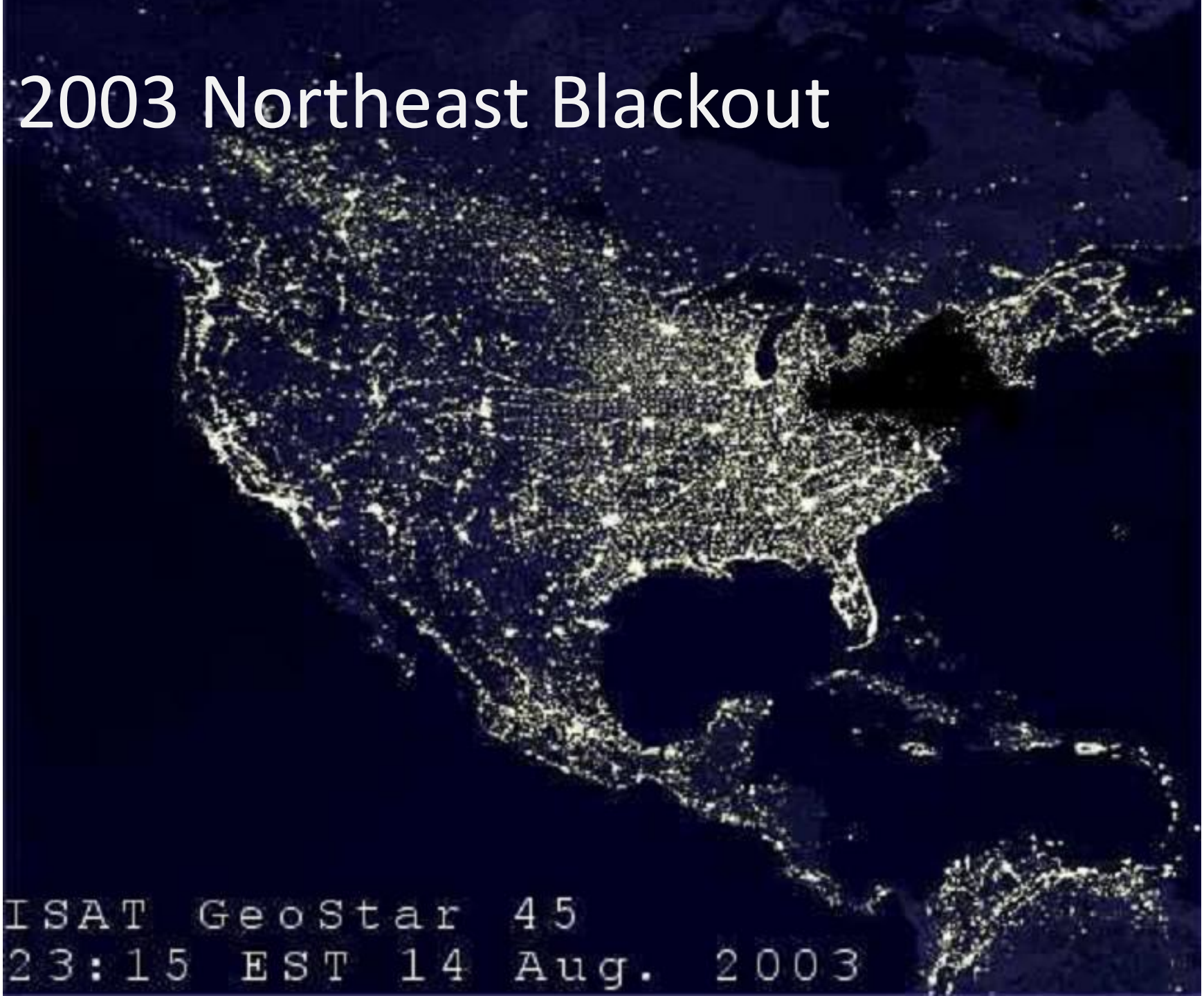
As I speak, there is feverish testing across the planet

- Low-wage test-hands are clicking through scripts
- Programmers are writing JUnit cases
- Elite testers are probing for weak points
- Some projects have continuous automated fuzz testing
- …

This is good…

- …but the world is getting more complex, more interconnected, and more software-dependent.
- We’re seeing the costs of that complexity

2003 Northeast Blackout



- August 14 2003
- Early afternoon
- Previously unknown fault causes power network alarms to be disabled – silently

- Now, what's a power distribution engineer's worst enemy?

- Unprocessed events queued up in alarm system
  - Eventually fails under the load
  - Backup alarm server starts up…
  - …then immediately fails too for the same reason
- Operators did nothing, because they perceived no problem
- Blackout cascades
- At peak, 50 million people in US and Canada without power
  - Took 4 days to fully restore function

Could they have found the fault?

- Race condition
  - Two processes could write to the same data at the same time
  - Caused alarm program to infinite loop
  - Very, very hard to trigger

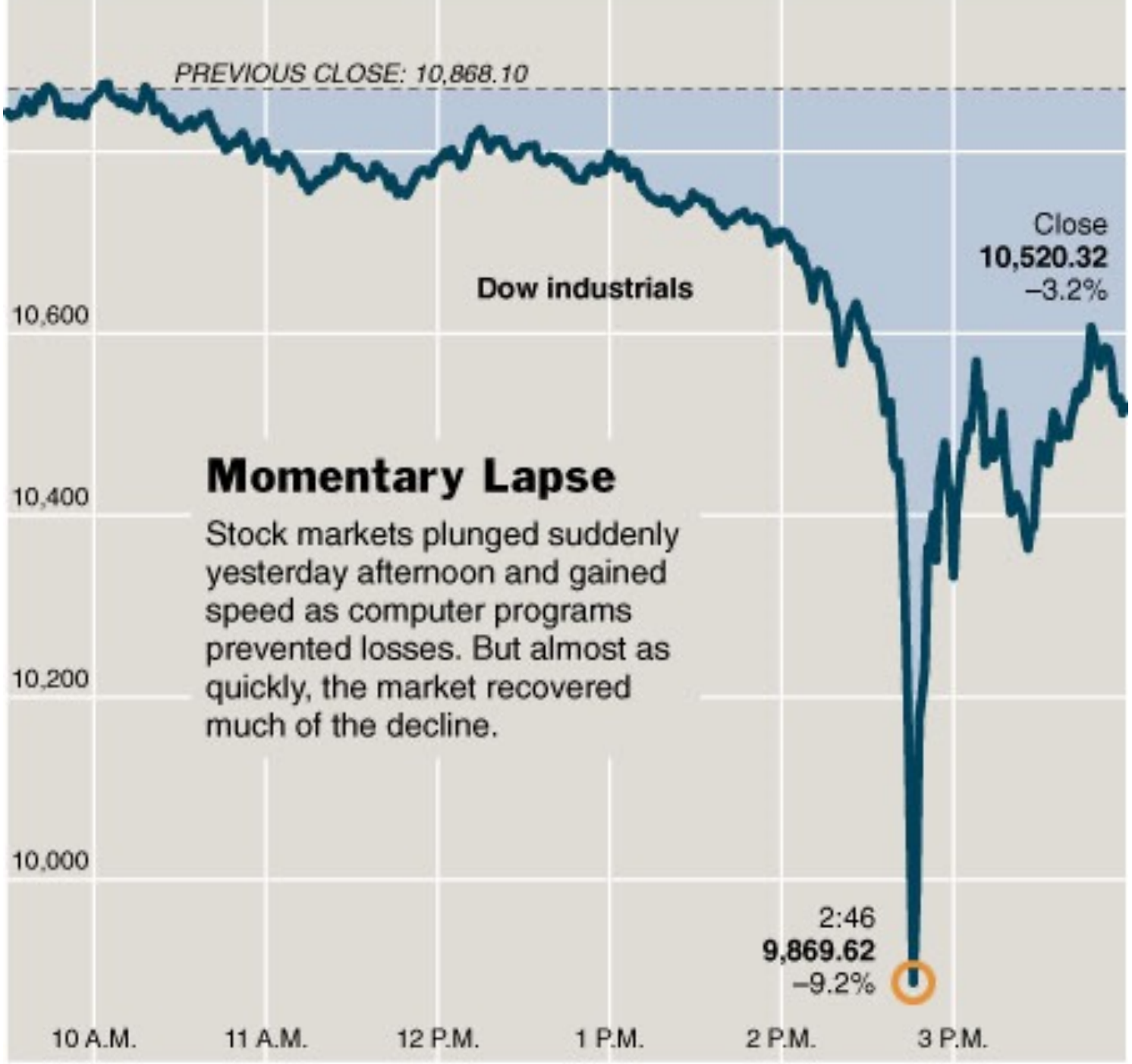
- Took 8 weeks to find ***after the blackout***

- *"We had in excess of three million online operational hours in which nothing had ever exercised that bug. I'm not sure that more testing would have revealed it."*
  - GE Energy's Mike Unum
  - (note how he says "more testing", not "better testing")

The Flash Crash

Flash Crash

- May 6, 2010
- 2:42 – 2:47 PM, the Dow Jones Industrial Average falls 600 points.
- Surprising? Overall fall for the day was about 1000 points – and that was the biggest one-day decline in its history.
- Fortunately, by end of trading the fall had been mostly reversed
  - What, tho, if this had happened at very end of trading day?


PREVIOUS CLOSE: 10,868.10
Dow industrials
Close
10,520.32
–3.2%
10,600
10,400
10,200
10,000
Momentary Lapse
Stock markets plunged suddenly yesterday afternoon and gained speed as computer programs prevented losses. But almost as quickly, the market recovered much of the decline.
2:46
9,869.62
–9.2%
10 A.M.
11 A.M.
12 P.M.
1 P.M.
2 P.M.
3 P.M.
Source: Bloomberg
THE NEW YORK TIMES

Flash Crash – the Causes

- Confusing, disputed, but it's clear that High Frequency Trading (HFT) was involved
  - Automated trading algorithms following pre-assigned rules

Flash Crash – Brief Narrative

- Early in day – a mutual fund sells a lot of “E-Mini” contracts
- Ordinary buyers are exhausted – all that’s left are HFTs
- HFTs start buying, then selling, then buying
  - Each time they’re selling for less and less
  - Price spirals downwards
- HFTs eventually shut down and stop trading
- At 2:45:28 PM, emergency “Stop Logic” function pauses E-Mini trading for 5 seconds
  - Prices stabilise, then rally

Where are the faults here?

- Lecture 1 faults were pretty foolish
  - Embarrassing that they weren't caught and fixed
  - Could have been atrocious (e.g. air traffic one)
- Northeast Blackout fault much harder to find
- And what even *was* the fault(s) in the Flash Crash?

Software faults don’t kill people… no, wait, they do

Therac 25

Therac 25

- Radiotherapy machine for cancer
- Two modes:
  - Electron
    - Weak beam, weak filter
  - X-ray
    - Strong beam, strong filter
  - At mode switch, turntable rotates to align correct filter with the (fixed-position) beam from the accelerator
- Turntable controlled by software

Therac 25 – the fault

- Turns out there's a fault
  - If user types quickly, they can access a third mode
    - Strong beam, no filter
    - Approx 100 times intended radiation dose
    - Let's call this "death ray mode"
  - User gets an error message, but it makes no sense and can be dismissed
- 6 patients seriously overdosed - estimated 3 died in consequence.

Given what I've said so far (complexity, fatality), what can we conclude?

We need to get better at testing

Trouble is, people often *don't* test well

- Key testing activities get pushed to the end, then dropped
  - E.g. whole-flight simulation
- Benchmarks look great – but other issues ignored
  - E.g. linux kernel and GUI responsiveness
- Bad news squashed by management
  - E.g. serious faults close to shipping

The solution?

- Case by case. This stuff is difficult.
- You can advocate for better testing
  - Especially if you're a consultant or high-status employee

Some Definite Non-Solutions

- Become an integrity zealot
  - Test everything like it's nuclear-launch capable
- Become a one-technique fanatic
  - *"Give me MC/DC or give me death!"*

- Doing those will:
  - Lead you to bad decisions
  - Damage your credibility
  - Damage the reputation of valuable techniques

A compromise solution

- Know a wide range of techniques
- Know many ways to think about those techniques
- Be flexible…
  - …just not to the point of irrelevance

We've been using a simple model of types of testing

- Dynamic / static testing
- Black / white / grey –box
- Functional / non-functional

- **Testers** – *who does the testing?*
- **Coverage** – *what gets tested?*
- **Potential problems** – *what risks are you testing for?*
- **Activities** – *how do you do the tests?*
- **Evaluation** – *how do you tell if a test passed or failed?*

Testers – *who does the testing?*

- In-house test team
- Realistic end users
- Subject matter expert
- …

Activities – *how do you do the tests?*

- Regression
- Manual scripted
- Exploratory
- Installation
- Load testing
- Long sequences

The Kaner classification is actually pretty complicated

- IEEE 829 (the software testing documentation standard) is much bigger
- …but you can understand them, in time

Kaner et al, Lesson 54

- “The Classification of a Technique Depends on How You Think About It”

Sometimes advocacy fails

- …because the commercial incentives are just too strong
  - Time-to-market can trump quality
  - Quality improvement might *cost* you market share
    - Especially if customers can't see your higher quality — "market for lemons"
  - N.B. this subject is called "Testing Economics" or "Quality Economics"
  - Best reference is probably Anderson's *"Security Engineering"* Ch 7

Regulation and engineering standards may make a difference

- E.g. DO-178B mandates testing and analysis for avionics software
  - MC/DC testing (at level A)
  - Statement coverage (at level A-C)
  - Documented traceability of tests to code, design and requirements
  - Any tools used must be *qualified* with regulator

Standards can be “quality theatre”

- How much does MC/DC actually guarantee?
- Standards + brains may be useful
  - Standards can *allow* testers to justify rigorous testing

- Good luck all on your exam (more on that to follow).
- Key message: don’t panic.
- These are unusual times.
- The exam and its marking will take into account the stresses that everyone is under!
  - Interpret this as generously as you can.