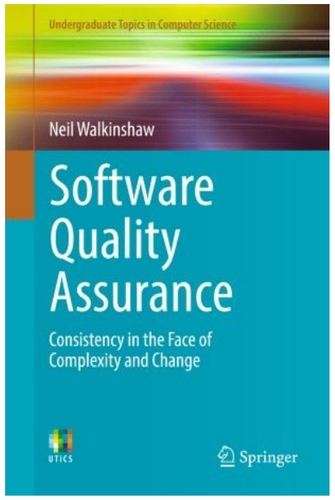
COM4506/6506: Testing and Verification in Safety Critical Systems

Dr Ramsay Taylor

Course history



Gordon Fraser

then Neil Walkinshaw

Course history

… now me!

“They used [my COM4506 assignment submission] as proof of my competencies and really liked it. Knowing about requirements traceability and the V model and stuff was incredibly useful and I was working on projects from week one rather than the month it normally takes for competency documents to be accepted.”

- James Beck, COM4506 2020/21,

  now Safety Critical Software Business Analyst at Arthur D. Little

Course Delivery

Currently booked:

- These lectures (Tuesday)
- “Computer Lab” (Wednesday 9am)

But it worked much better as “flipped learning” last year.

Assessment

30% Assignment (Week 9 - 12)

- Somewhat technical - Java/C and Software Testing
- Somewhat Analytical - A report, not just some code

70% Exam

- Probably “online” so “open book”
- Similarly analytical

You said…we did:

Question	Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree	%Favourable
It is clear how this module fits within my programme as a whole.	0.0%	0.0%	0.0%	100.0%	0.0%	100.0%
The teaching was effective in helping me learn.	0.0%	0.0%	16.7%	50.0%	33.3%	83.3%
The teaching staff were approachable.	0.0%	0.0%	33.3%	33.3%	33.3%	66.7%
During this module I have had opportunities to reflect on my progress.	0.0%	0.0%	50.0%	50.0%	0.0%	50.0%

So I will add some *formative* quizzes on Blackboard (which may or may not work!)

What is this actually about?

Testing, Verification, Safety, Critical Systems, and stuff...

- Some technical stuff
  - Test generation
  - Test metrics
  - Specification languages
- Some analytical/ “Engineering” stuff
  - Risk/Hazard analysis
  - Development processes for standard compliance
  - Safety Cases

Course outline

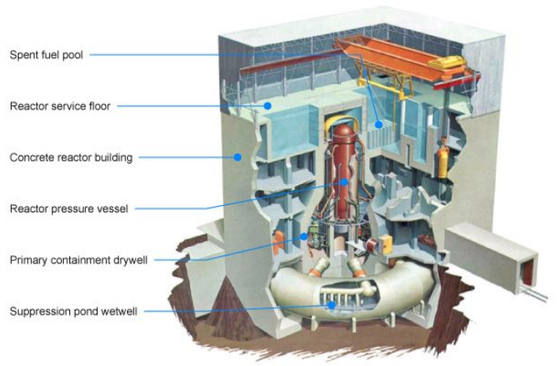
- Safety-Critical Systems
  - What are they?
  - How do we classify them?
  - How do we identify them?
- Specifications
  - What is the system actually supposed to (not) do?
  - Specification languages
  - Specification level verification
- Implementation
  - Correct by Construction?
  - Assertions
  - Coding Standards
- Testing and Verification
  - Software Testing
  - Systems/Integration Testing
  - Static Analysis
  - Formal Verification
  - Test Adequacy - Testing the Tests
- Engineering Processes for Critical Systems
  - Where does this fit in to development processes
  - Industry Standards (more interesting than it sounds, I promise!)

Safety Critical Systems

A system where failure or malfunction may result in:

- *death or serious injury to people*
- *loss or severe damage of equipment/property*
- *environmental harm*

Safety Critical Systems



Many safety issues can be reduced or *mitigated* by design outside of the software

Safety Critical Systems

Many safety issues can be reduced or *mitigated* by design outside of the software

…but not all of them!

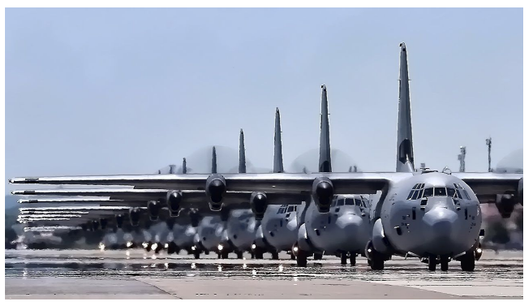
Safety Critical Systems

"Design it so that it never kills anyone"

"Fail safe"

Development processes

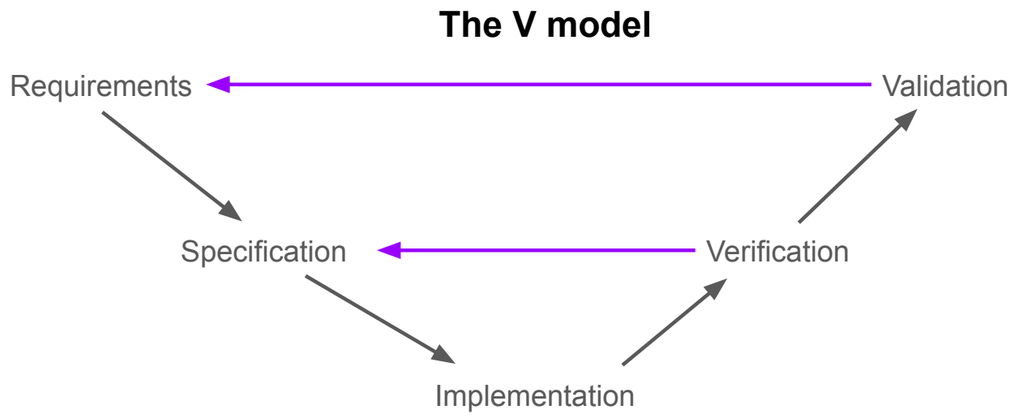
- You have seen some Software Development processes already (???)
- The relevant standards include some process requirements, and some steps that are required (or a lot easier!) at different points in the process



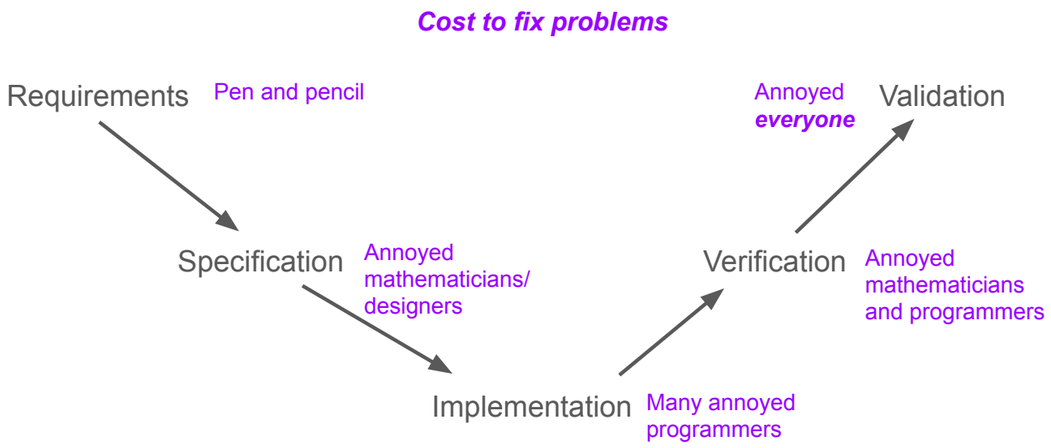
Nice plane, we'll take 30 of them.

You did conform to all of our current national airworthiness standards 67 years ago when you were designing them, right?

Verification and Testings



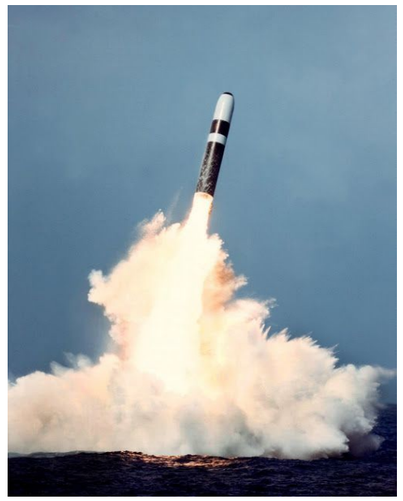
Verification and Testings



Maintenance, “Improvements”, COTS

- Once a system is “in service” it will still be developing.
- Complete, well defined systems can be reused in different settings (Commercial Off The Shelf (COTS))
- Sometimes problems are created with the best of intentions!
- This is yet another reason to make the Requirements and Specification well defined!

What about Agile Development?



“Move fast and break things”

“Can we *pivot* into Fintech?”

“The code *is* the documentation!”

I’m **not** criticising Agile per-se!

Safety Critical system are constrained by other Engineering disciplines.

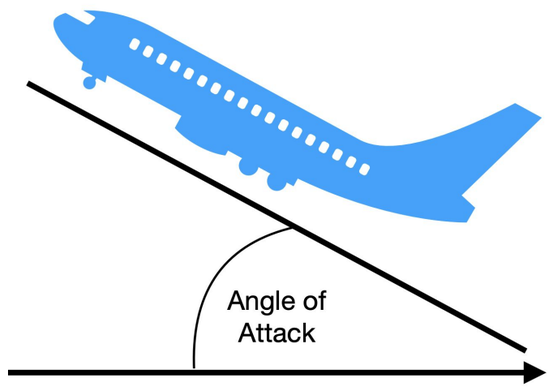
E.g.: 737-MAX MCAS

Different engines from other 737s.

Boeing identified awkward pitching in some circumstances.

Developed Manoeuvring Characteristics Augmentation System (MCAS) to counter this: Automated nose-down procedure if either sensor indicates high AOA, reactivates every 5 seconds.

Could be deactivated by the pilot if necessary (if you know how...)



Safe Design in Use

You have two altimeters, that are fairly reliable. How high are you?

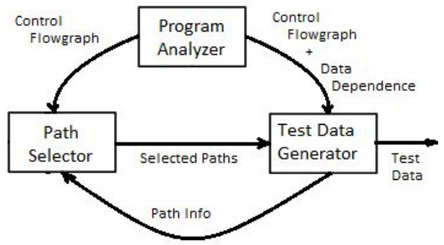
Only wrong 0.01%
of the time

Only wrong 0.01%
of the time

AERO students: ignore the difference in the pressure setting, that isn't the point I'm trying to make!!

What has that got to do with Testing or Verification?

- We will discuss "classic" software testing (Unit testing, test cases, corner cases, etc.)
- Critical system design ought to include *Properties* (safety, or security, or whatever) that can drive testing and verification.



Safe Design in Use

You have **three** altimeters, that are fairly reliable. How high are you?

Only wrong 0.01%
of the time

Only wrong 0.01%
of the time

Only wrong 0.01%
of the time

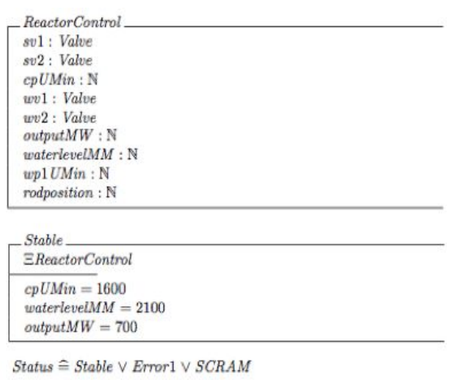
What has that got to do with Testing or Verification?

- Evaluating (and convincing someone else!) how well you've tested or verified software is important.
- Evaluating your testing *against some Requirements* is important.
- Human error and software error can exist in the Tests as well!

```
state_loop(ScanTasks, NextScanId, Playbacks, NextPlaybackId, Dict) ->
    receive
        {'DOWN', _MonitorRef, process, ScanTask, _Reason} ->
            case lists:keytake(ScanTask, 2, ScanTasks) of
                {value, {ScanId, ScanTask}, NewScanTasks} ->
                    ?INFO("Removing Sat>Ip scan id ~p from state", [ScanId]),
                    state_loop(
                      NewScanTasks, NextScanId, Playbacks, NextPlaybackId, Dict);
                false ->
                    state_loop(
                      ScanTasks, NextScanId, Playbacks, NextPlaybackId, Dict)
            end;
        {From, get_scan_tasks} ->
            From ! {?MODULE, scan_tasks, ScanTasks},
            state_loop(ScanTasks, NextScanId, Playbacks, NextPlaybackId, Dict);
        {From, create_scan_task, ScanTask} ->
            erlang:monitor(process, ScanTask),
            From ! {?MODULE, created_task, NextScanId},
            state_loop(
              [{NextScanId, ScanTask} | ScanTasks], NextScanId + 1,
              Playbacks, NextPlaybackId, Dict);
        {From, get_playbacks} ->
            From ! {?MODULE, playbacks, Playbacks},
            state_loop(ScanTasks, NextScanId, Playbacks, NextPlaybackId, Dict);
        {From, GetOrTake, PlaybackId}
        when GetOrTake =:= get_playback;
             GetOrTake =:= take_playback ->
            {Playback, NewPlaybacks} =
                case lists:keytake(PlaybackId, 1, Playbacks) of
                    {value, {PlaybackId, P}, N} ->
                        {P, N};
                    false ->
                        {undefined, Playbacks}
                end,
            From ! {?MODULE, playback, Playback},
            state_loop(
              ScanTasks,
              NextScanId,
              update_playbacks(GetOrTake, Playbacks, NewPlaybacks),
              NextPlaybackId,
              Dict);
        {From, add_playback, Playback} ->
            From ! {?MODULE, added_playback, NextPlaybackId},
```

What has that got to do with Testing or Verification?

- A lot of our ability to Test (and verify) depends on having *well defined* Specifications, and Properties.
- *Ambiguity* can be the source of problems throughout.



Now what?

[Assuming we decided on flipped learning:] Some video lectures will come out before next Tuesday... Watch them!

We'll do something more interactive in this slot next week.