COM4506/6506: Testing and Verification in Safety Critical Systems

Dr Ramsay Taylor

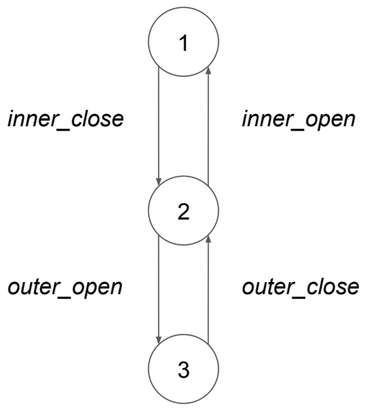
Contents

- Model Based Testing
- Model Inference as a feedback system
- Test set minimisation

Test Generation

Test Generation

Model Based Testing

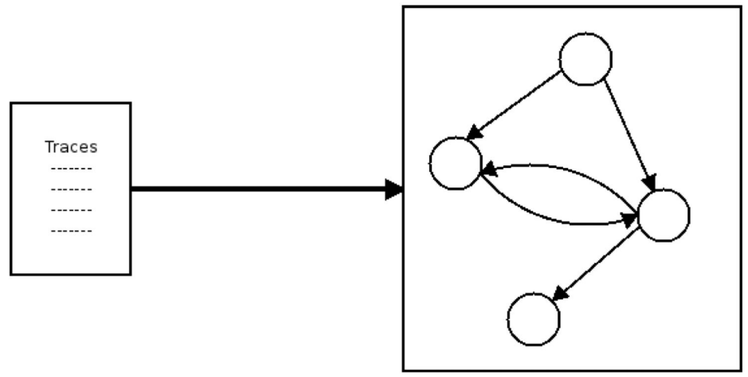


Models from the spec can be a source of (automated) tests - if they are formal enough!

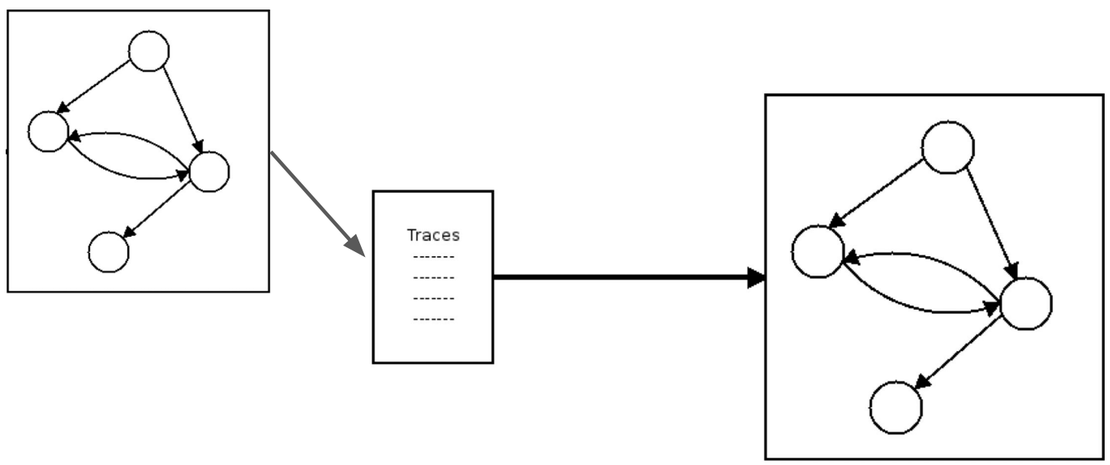
We did verifications on traces, for example. Traces can become tests:

traces(P) = {<>, <inner_close>, <inner_close, outer_open>...}

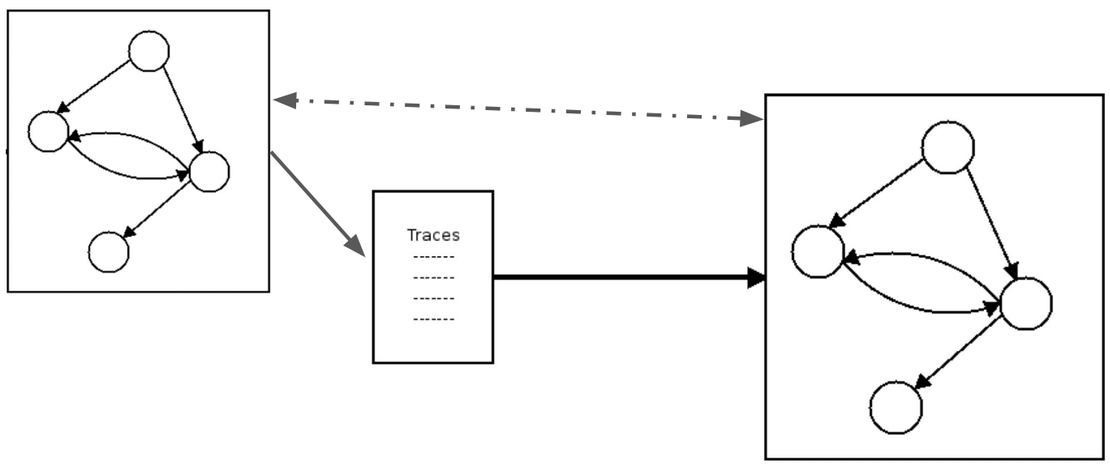
Model Based Test Testing …



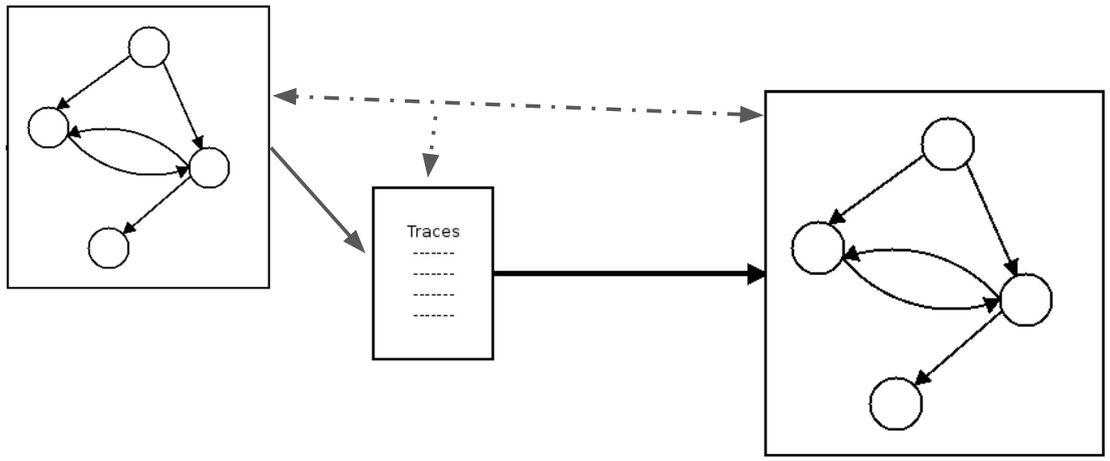
Model Based Test Testing …



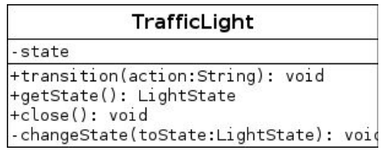
Model Based Test Testing …



Model Based Test Testing …



API Testing



The external interface (and possibly some of the internals) will be in the detailed software design spec.

We can *explore* this in various structured ways.

Reflection

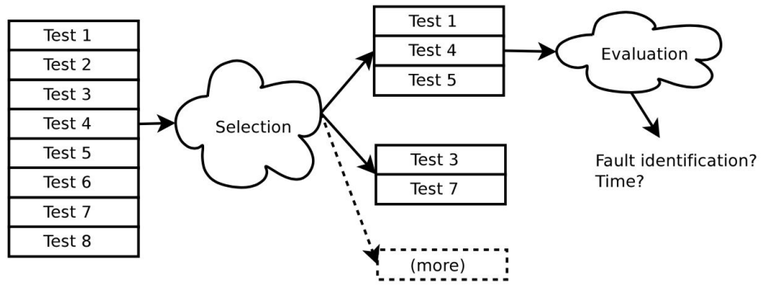
Some languages allow you to access details of the program from within programs…

This lets us write automated tools to *discover* as well as *explore* the interface.

This is usually a bad thing in security critical systems! Incorporating Reflection can make it incredibly difficulty to really assure Safety Critical systems too...

Test Set Minimisation

If we make 1000 tests and it makes a good test set, do we still need all 1000?



Test Set Minimisation

What do we even mean by ***good****?*

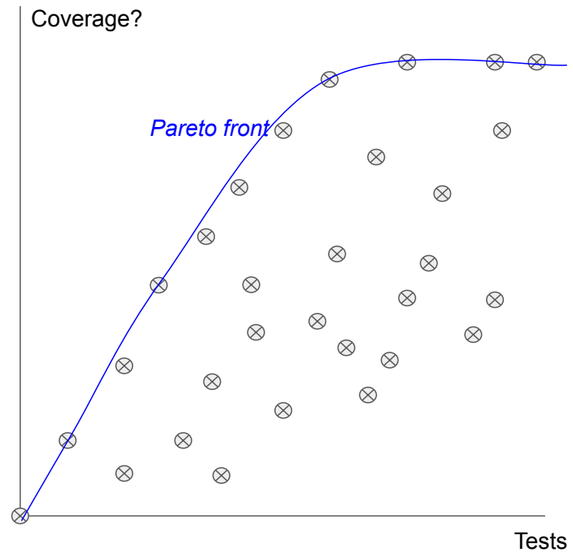
Fast?

Maximum coverage?

Both?

Kinda both but more fast if still mostly covering…

...or as fast as possible for 90% coverage...



Summary

- If we have *Structured Documents* for our Spec we can *automate* some test generation
- We can also do that with *Language Reflection* (or some static analysis, symbol tables, etc. etc)
- We can also *Minimise* our test sets - if we know what our priorities are!