Testing RAVEN

UNIVERSITY OF ICELAND www.hi.is

Helmut Neukirchen

Faculty of Industrial Engineering, Mechanical Engineering and Computer Science

University of Iceland, Reykjavík, Iceland

helmut@hi.is

About me

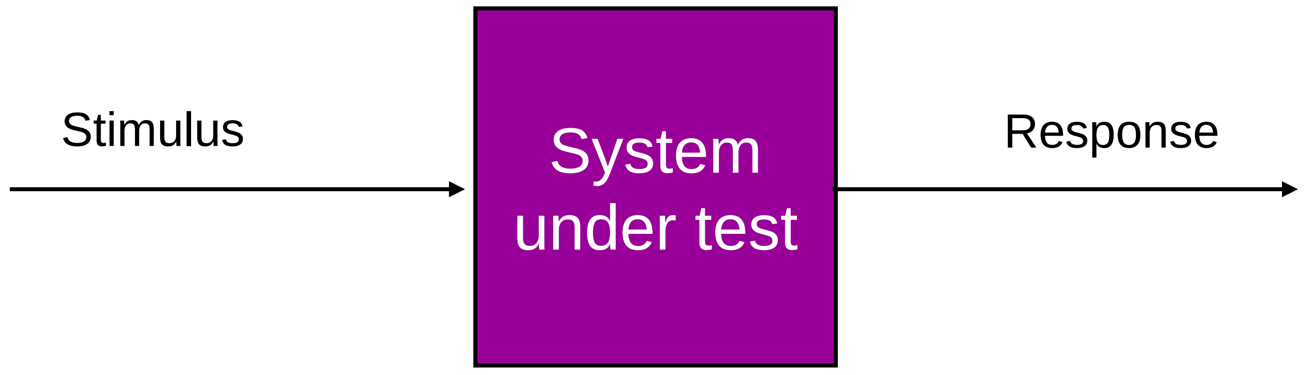
- Helmut Neukirchen
  - Associate Professor for Computer Science at the University of Iceland since 2008.
    - Currently in parental leave until end of 2010
  - Before: researcher and lecturer in Germany.
    - Phd in Computer Science & Postdoc at University of Göttingen
    - Started PhD studies at University of Lübeck
    - Studied Computer science at RWTH Aachen
- Research interests: Software Engineering & Distributed Systems
  - Software Quality, Testing Distributed System, Grid computing & Cloud computing.

What is Software Testing?

- "*Testing is the process of executing a program with the intent of finding errors*." (Myers)
- "*Program testing can be used to show the presence of bugs, but never to show their absence!*" (Dijkstra)
- Testing is an expensive (up to 50% project costs), but important means of quality management.
  - Other means: reviews, checking of models, etc.

How does testing work?

- Stimulus is sent to system under test.
- Observe response.



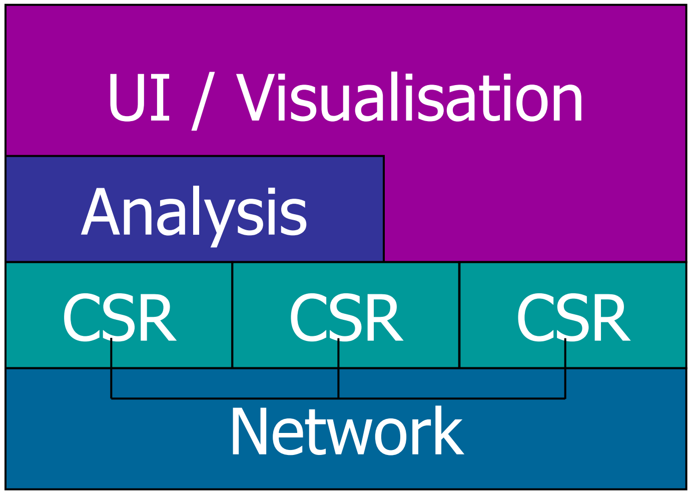
Test types

- Functional test:
  - Test passed as long as observed response fits stimulus.
- Non-functional test:
  - Performance:
    - Response must not only be correct, but be observed within certain time-limits.
  - Scalability/Load:
    - Test correct & timely response at different system sizes & work loads.
  - Robustness
  - Security
  - Usability
  - …

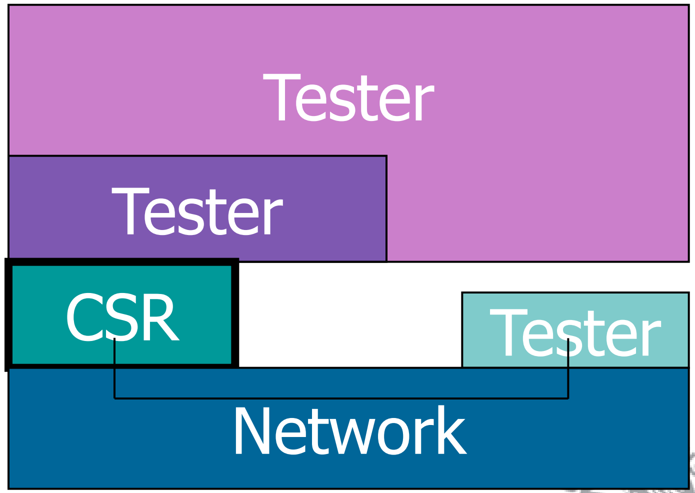
Test levels: Unit test

- Test a single class by calling a method, test a protocol layer by sending a network message.
  - Cover methods/messages.

RAVEN System:



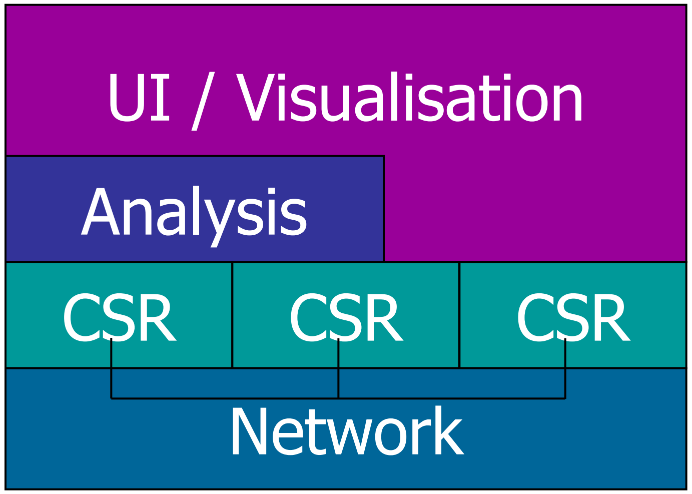
Unit test of RAVEN CSR Unit:



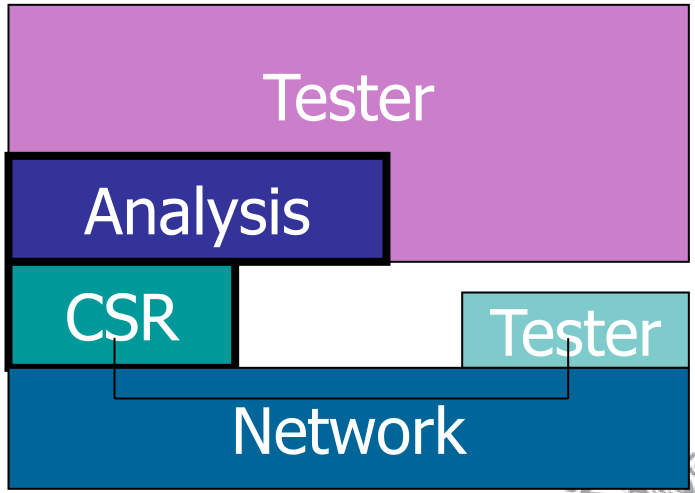
Test levels: Integration test

- Test that multiple units are able to work together.
  - Cover interface between integrated units.

RAVEN System:



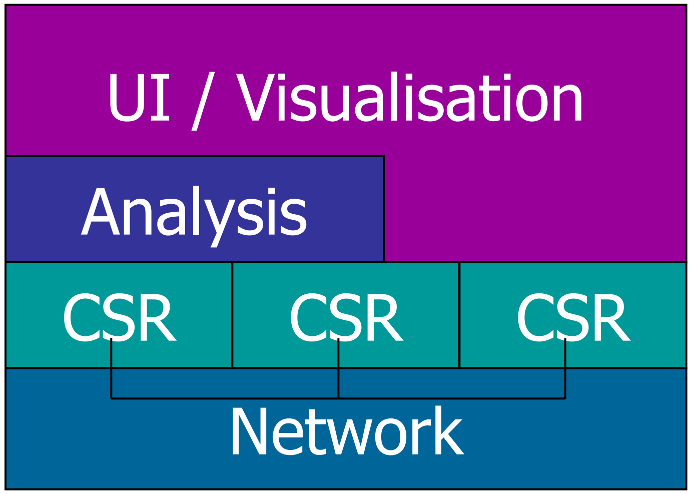
Integration test of RAVEN Analysis interacting with RAVEN CSR:



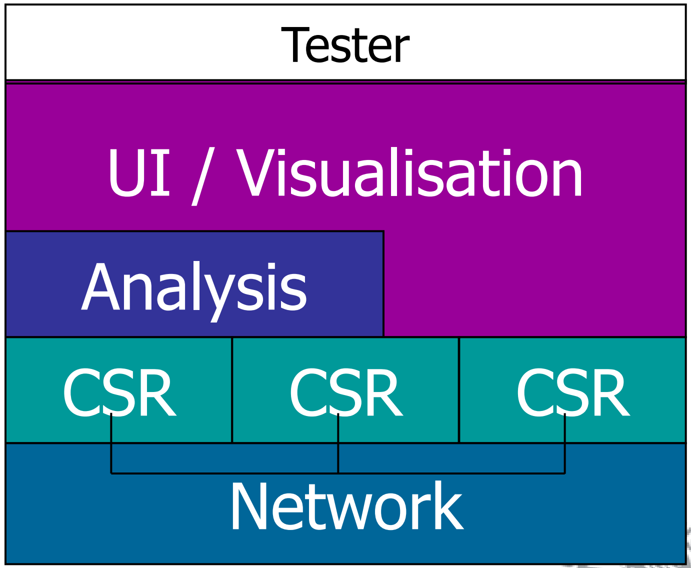
Test levels: System test/Acceptance test

- Test the system as a whole via it's user interface.
  - Cover usage scenarios.
  - System test: system tested in an artificial test environment.
  - Acceptance test: system tested in final environment.

RAVEN System:



System test of RAVEN:



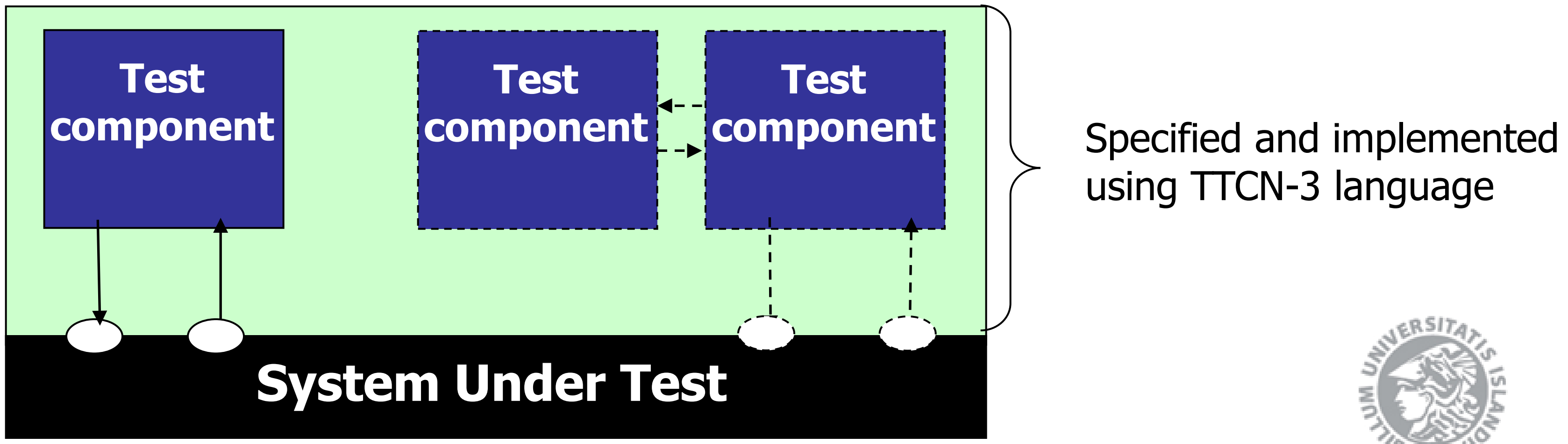
Automated test execution with TTCN-3

- Testing and Test Control Notation version 3
- Language for specifying distributed tests.
- Compilable into executable code: automated test execution.
- Standardised technology:
  - European Telecommunication Standards Institute (ETSI).
  - International Telecommunication Union (ITU).
- Strength in functional testing of distributed systems.
  - However, applicable to other domains, levels, and types of tests:
    - From unit test to acceptance test level.
    - From functional to non-functional (load, performance) tests.
    - 3G/UMTS networks, IP networks, Grid.

Rings, Neukirchen, Grabowski: Testing Grid Application Workflows Using TTCN-3. International Conference on Software Testing, Verification, and Validation, IEEE 2008

Distributed testing with TTCN-3

- Testing a distributed system often requires a tester that is distributed itself.
- TTCN-3 supports this by the notion of test components:
  - execute behaviour of a test,
  - may run on different nodes of a test environment,
  - may co-ordinate their behaviour.



Testing RAVEN

- Functional test of RAVEN (at all levels: unit test, integration test, system test).
  - Networking/communication aspects:
    - TTCN-3-based standard approach from protocol testing for telecommunication and network systems.
  - UI/visualisation aspects:
    - Standard capture/replay approach may not work here.
  - Analysis aspects:
    - Feasible as long as small “toy” examples are used as test data.

Challenges of testing RAVEN

- Performance test, Scalability/Load test of RAVEN:
  - Heisenberg's uncertainty problem at the software (& hardware) scale: by observing (= testing) a system, we influence it, e.g.
    - Communication overhead to co-ordinate distributed test components reduces network capacity.
    - Self monitoring ($\rightarrow$visualisation) overhead.
  - How to test analysis algorithms working on PB of test data?
    - How to manage (generate, store) PB of test data?

  $\Rightarrow$ Only possible to a certain size within an artificial test environment.
    - Performance & Scalability beyond this size only testable by productive use in real environment?
    - Performance & Scalability beyond this size may evaluated by simulation based on performance models!
    - How do others (e.g. Apache Hadoop) do this?
      - A quick look: small artificial tests, big productive-use test.