AARHUS UNIVERSITET

Microservices and DevOps

DevOps and Container Technology

Testability: An Architectural Quality Attribute

Henrik Bærbak Christensen

Testability

- **Testability**: Concerned with the ease with which the software can be made to demonstrate its faults

- Thus an estimate of *effort* (to make test) and *efficiency* (probability of finding a failure)

Testability

- **Testability**: Concerned with the ease with which the software can be made to demonstrate its faults
- Techniques:
  - **Testing:**

    Definition: **Testing**

    Testing is the process of executing software in order to find failures.

  - Review
    - Manual: Structured and systematic human *reading* of programs
    - Static analysis: let programs analyze your program
  - Formal verification: make profs that you program works

AARHUS UNIVERSITET

Failure and Defects

- What we observe when testing

> **Definition: Failure**
>
> A failure is a situation in which the behavior of the executing software deviates from what is expected.

- Why we observe it – the cause

> **Definition: Defect**
>
> A defect is the algorithmic cause of a failure: some code logic that is incorrectly implemented.

- På dansk: Fejl og fejl ☺

AARHUS UNIVERSITET

Terminology

- Test Case

> **Definition: Test case**
>
> A test case is a definition of input values and expected output values for a unit under test.

(input, output, unit under test)

- Which means:
  - We have to isolate some part of the software – the *'unit'*
  - We have to be able to *provide input* to the unit
  - We have to be able to *execute the unit with the input and observe the output (which requires a specific context)*
  - We have to know what *output to expect (oracle)*

Test Case

- Manual Testing

> **Definition: Manual testing**
> Manual testing is a process in which suites of test cases are executed and verified manually by humans.

- Automated Testing

> **Definition: Automated testing**
> Automated testing is a process in which test suites are executed and verified automatically by computer programs.

- Trend: Towards automated
  - Netflix, Uber, MS, Google, …

AARHUS UNIVERSITET

… Can be difficult!

- Because: It is difficult or tedious to *provide input*
  - Start a web server with five dependent systems
    - Ensure that the two databases used are in a correct state,
      - One is that Arne is a registered user in the user database
      - Another that Arne's current balance on his account is 200€
  - Log in Arne from the web page
  - Go to the account page and enter (300€, Birte) in the 'transfer funds' page.
  - Validate that transfer is refused and the message is 'out of funds'

AARHUS UNIVERSITET

… Can be difficult!

- Because: It is difficult or tedious to *provide input*
  - Start a web server with five depe
    - Ensure that the two databases u
      - One is that Arne is a registered
      - Another that Arne's current bala
  - Log in Arne from the web pa
  - Go to the account page and funds' page.
  - Validate that transfer is refused an

Tedious to set *object's state* to well defined values before executing the test.
(And resetting is hell!)

Tedious to enter the *input parameters* of the test case.

Tedious to verify expected output match computed output

AARHUS UNIVERSITET

… Can be difficult!

- Because: Difficult to *execute unit in isolation*
  - If unit deeply nested inside a complex system
    - Impossible/difficult to isolate
    - Difficult to control surrounding units

- Anti decomposition axiom:
  - ”You cannot fully test a module through testing the system”
- Anti composition axiom:
  - ”You cannot fully test a system through testing all units”

AARHUS UNIVERSITET

… Can be difficult!

- Because: It is difficult to *get the output*
  - If the thing we need to validate is printed in a mail that the system sends to a user
    - Have to log in as this user, open mail box, verify contents
  - If the proper answer is that the graphite rods are fully extracted from our nuclear core
  - If the proper answer is that 250 states change in 125 different systems

AARHUS UNIVERSITET

… Can be difficult!

- Because: Given we have the output, what is the *correct expected output*?
  - Big legacy systems tend to do stuff which we knew why happens – a decade ago…
  - And user rely on what *it has always been doing*, not what it was specified to do!
- War story:
  - use the algorithm itself to compute the answer to expect

… Can be difficult

- Because: Any change potentially require regression testing!

> **Definition: Regression testing**
>
> Regression testing is the repeated execution of test suites to ensure they still pass and the system does not fail after a modification.

- Let us face it:
  - It is expensive so either
    - Our product becomes too expensive
    - We just hope for the best
- WarStory: The 1.000 hour manual test system…

AARHUS UNIVERSITET

Testing Issues in Summary

- Definition: **The Testability Input Issue**
  - Embody the issues involved in providing comprehensive and deterministic input to the unit under test in a reliable and reproducible way
- Definition: **The Testability Unit Isolation Issue**
  - Embody the issues involved in testing a unit under test in isolation in a comprehensive environment in a reliable and reproducible way
- Definition: **The Testability Output issue**
  - Embody the issues involved in recording the output from a unit under test and asserting the correctness in a reliable and reproducible way

AARHUS UNIVERSITET

Testability Tactics

Tactic

- **Tactics**: Architectural techniques to achieve required quality attributes
  - i.e. control the response measure in a positive direction

- So
  - Architectural techniques to increase testability

AARHUS UNIVERSITET

Testability Tactics

Figure 10.4. Testability tactics

Tactics

- Control/observe system state
  - Specialized interfaces
    - Encapsulation works against validating intermediate results
    - Compare Spy's retrieval interface
  - Record/playback
    - Record interaction at interface boundary for later playback
    - Many web testing tools (Selenium a.o.) work this way
  - Localize state storage
    - To enable testing when UUT is in particular state
      - Ex: Backgammon rules change at end of game, but tedious to get there if by moving one piece at a time
  - Abstract data sources
    - Make it easy to control UUTs input data
      - Stubs, *program to an interface* and *use delegation*

Tactics

- Control/observe system state
  - Sandbox
    - Isolate system from ‘real world’ to enable experimentation
      - Isolate from production data and env
      - Allow transactions to be easily rolled back
      - Use Virtualized resources / VMs
    - Use stubs, mocks, dependency injection for
      - Real clocks, real hardware, real sensors, real …
  - Executable assertions
    - Class level invariants, pre- and post-conditions
    - Checked continuously at run-time

Tactics

- Limit Complexity
  - Limit structural complexity
    - Make smaller and more cohesive abstractions
      - High cohesion, low coupling, separation of concern
    - Eventual consistency easier than always consistent
      - Simpler code and easier to test
  - Limit non-determinism
    - Avoid non-determinism as best possible
      - Stubbing randomness for instance

Evolving World

- Also here I find a tactic missing (or a category)

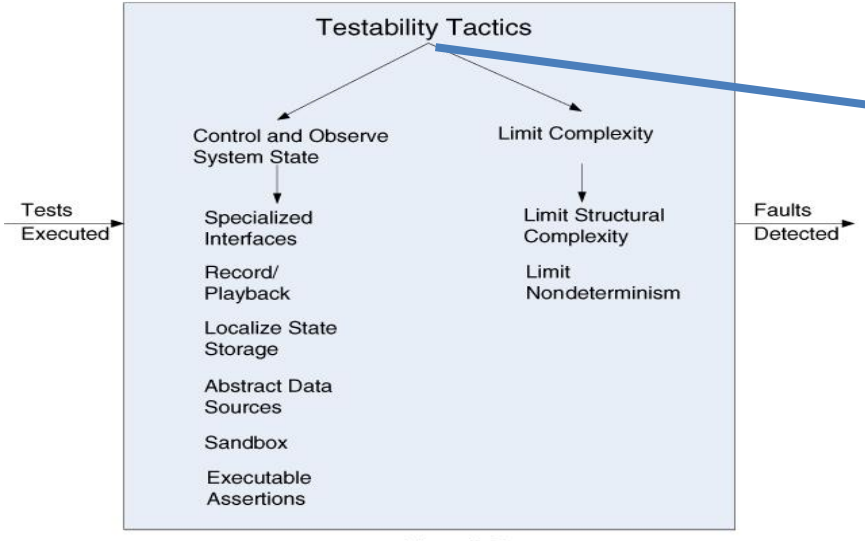


Figure 10.4. Testability tactics

'Ease of demonstrate faults' equals **speed!**

Test Process
- Automation
- RegressionSpeed
- Monitoring

Tactic

- Test Process Tactics
  - **Automation:** Ensure that tests are executed automatically/programmatically, not by hand
    - xUnit frameworks
    - Continuous Integration servers on dedicated branches
  - **Regression Speed:** Ensure your automated tests can be executed *fast*. Unit tests in seconds, integration/service tests in minutes, system/end-to-end tests in hours.
    - Service Doubles

Tactic

- Test Process Tactics
  - **Monitoring:** Monitor production systems and report anomalies
    - Monitor log messages
    - Monitor physical server farm health
    - Simian army to produce failure conditions in prod.

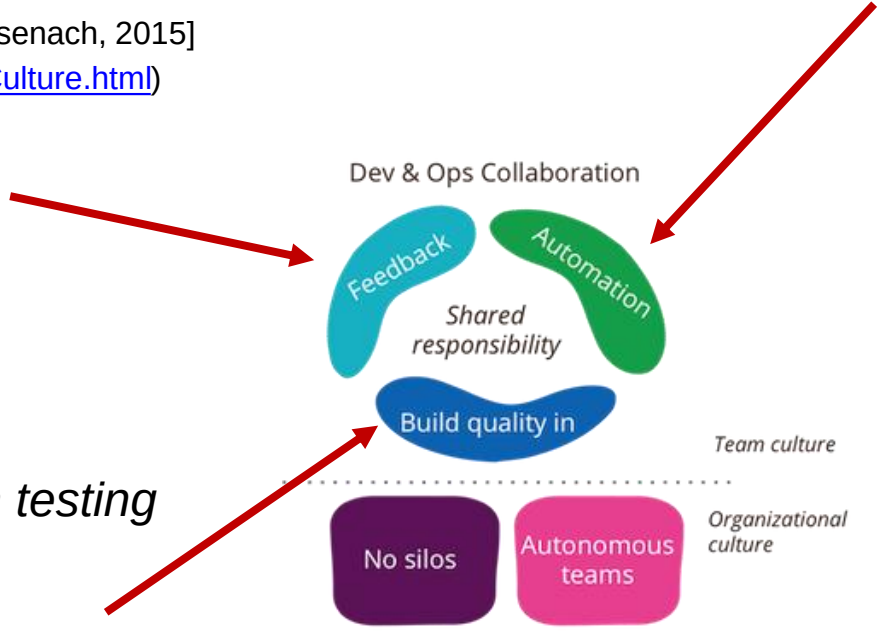
AARHUS UNIVERSITET

Testability and MSDO

I did not sign up for a test fagpakke, did I?

Yes you did ☺

- DevOps Culture [Rouan Wilsenach, 2015] (https://www.martinfowler.com/bliki/DevOpsCulture.html)
  - We need
    - Fast feedback
    - Quality Code
    - Automation
  - Main technique
    - *Automated regression testing*



AARHUS UNIVERSITET

So – in General

- All features/quality attributes should be *demonstrated through automated testing* in this course

- Write JUnit code to validate at unit testing level
  - Using test doubles to control indirect input and ouput

- Write JUnit+TestContainer code to validate at integration testing level
  - Use real-life containers to handle deterministic input and output
  - (And test double services or test doubles for non-determ.)