Testing tutorial

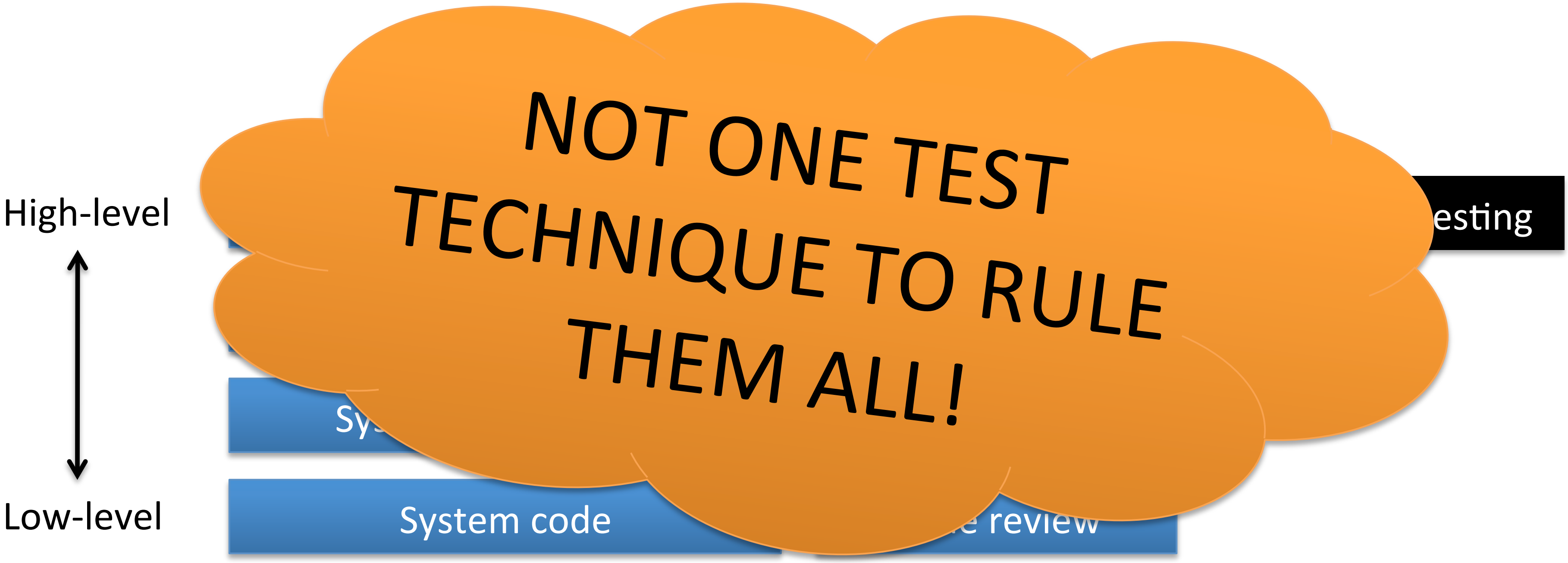
Agile Development Processes 2014

Eric Knauss and Emil Alégroth

Background to Testing

- Why do we test?
  - Check that the system under test (SUT) works!
  - Check that the requirements are fulfilled
  - Check that the SUT works as the customer expects
- Verification
  - Verify that the system works
  - Ex: Code review, unit testing, integration testing
- Validation
  - Verify that the system works according to the customer expectations
  - Ex: System testing and Acceptance testing

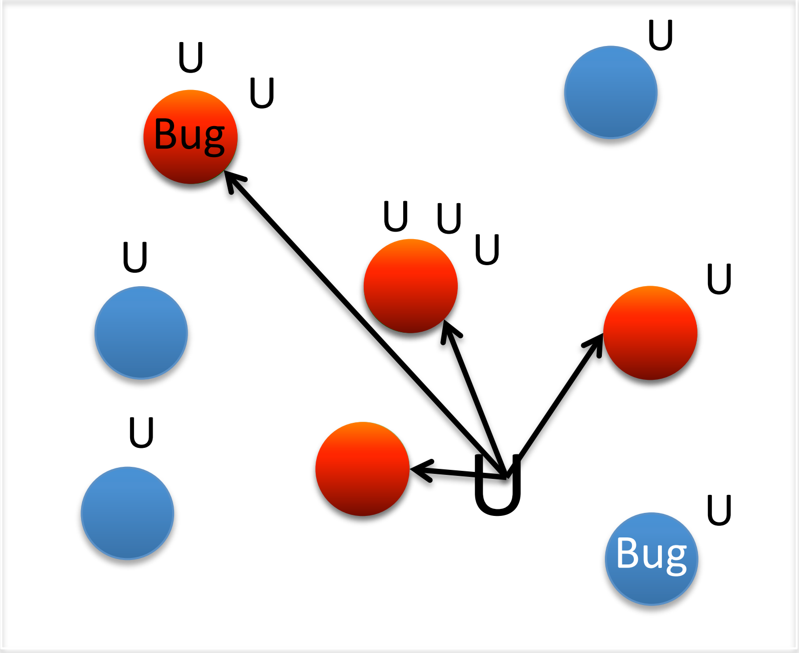
System representation



NOT ONE TEST TECHNIQUE TO RULE THEM ALL!

Automation

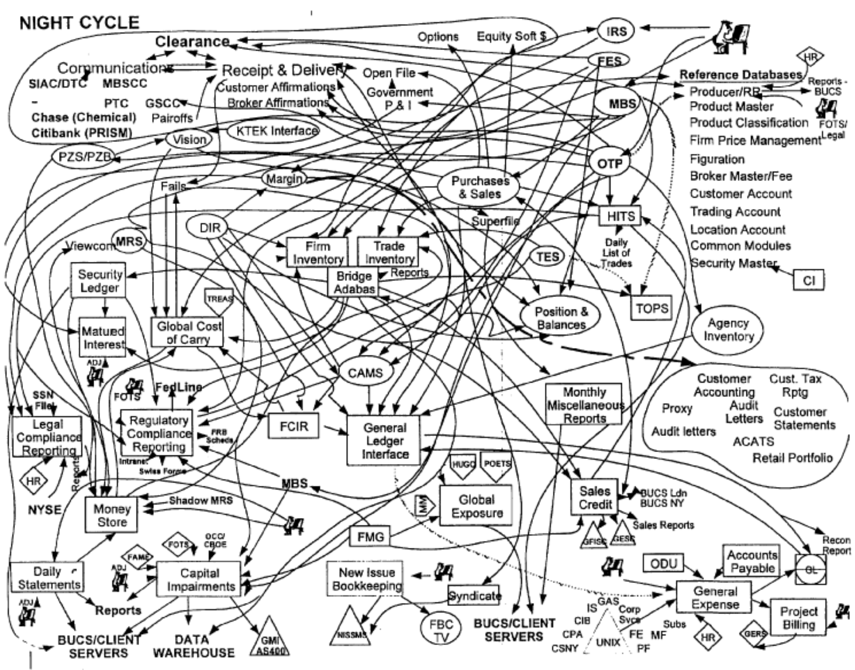
Unit tests (Low level testing)



Record and Replay (High level testing)

How do you perform effective testing?

- A good test process
  - Code review
  - Unit/Component tests
    - Commit code
  - Integration tests
  - System tests
    - Manual
    - Automated
  - Acceptance tests
    - Customer



Random picture! Not actual process!

- What practices are there for testing in Agile Development?

TestFirst

- If testing is good, then testing more often / always is even better
  - We want to embrace change – Regression testing

- Idea: Write test early, even before implementation

1. Write test
2. Let test fail
   - Do we really test non-existing functionality?
3. Implementing, until test is green
   - As ***simple as possible***!
4. Refactoring

Principle of TestFirst: a Dialogue

Task: Java method len(int) returns number of digits of an int.

Test starts
„len(5) should be 1!"

```
assertEquals(1,
    len(5));
```

JUnit

COMPILER-ERROR!
What is the meaning of "len"?

Program: That is easy:
public int len (int zahl) { return 1; }

JUnit: ok. Testcase fulfilled.

Test: Just you wait!
„len(321) should be 3!"

```
assertEquals(3,
    len(321));
```

JUnit: Error!
1 instead of 3

Program: No problem …
if zahl<10 then return 1 else return 3

JUnit: ok.

Test: I don't believe this!
"len(12345678) should be 8!"

```
assertEquals(8,len(12345678));
```

JUnit: Error!
3 instead of 8

Program: … ok, I see a pattern here:
for (i=…

Test-Driven Development

Testcases and automatic regression tests for every class in product

10 The automated tests are the design. The on-site customer makes the acceptance tests.

8 After doing design and prototypes, we create a few testcases

6 As soon as the code is done, we create thorough unit tests,
only after that goes the code to the test team.

4 We have heard about JUnit. Never tried it though.

2 Our system test phase always runs out of time: There are many errors!

0 We do not test explicitly. Sometimes a customer tells us when there is a problem.

c.f.: Krebs, William (2002):
Turning the Knobs: A Coaching
Pattern for XP through Agile
Metrics. Springer, Lecture Notes
on Computer Science 2418

?

- Are those tests Blackbox or Glassbox?

- Traditionally, programmers and testers are supposed to be different persons.
  - Why?
  - Does that not kill the testfirst idea?