What We Knew about Testing 10 Years Ago - and Still don't Do - and Why

Hans Schaefer
Software Test Consulting
N-5281 Valestrandsfossen
hans.schaefer@ieee.org

What did we know
What should be matters of fact
Why they are not

About myself

Lived in Germany until 1981
- **Working in IT since 1979 (civil engineer, inforr**
- **Process control software development**

QuickTime™ og en TIFF (LZW)-dekomprimerer kreves for å se dette bildet.

Moved to Norway
- **CASE tool development and test**
- **Test tool development and tailoring**
- **Consulting in Quality Assurance matters**
- **Independent consultant since 1987**
- **University lectures on software testing in Norway**
- **Consulting in Norway, teaching software testing abroad**

Non job: working with steam locomotives, glacier guide.
- **See my home page http://home.c2i.net/schaefer/**

Testing 1993 - 1994

Sources:
Myers, The Art of Software Testing, 1979
Beizer, Software Testing Techniques, 1990
Kaner et al., Testing Computer Software, 1988 (1999)
Hetzel, The Complete Guide to Software Testing, 1988
EuroSTAR Conference 1993, 1994

My jobs with Test Process Improvement in Companies

10 Important Matters of Fact

1 Early test preparation prevents defects
2 Systematic test case selection methods are already in place
3 The tester shall evaluate, not show that it works
4 Test coverage should be measured
5 Inspections are necessary
6 Testing should be automated
7 Before automation we need a systematic process
8 Developers should unit test!
9 Test cost depends on the quality of the system going into test
10 The V-model is useful.

1 Early test preparation prevents defects

To prepare a test gives feedback.
The tester thinks in a concrete way and asks concrete questions.
Even the test needs serious preparation.

Why don't we do it?
- Optimism: It probably works.
- Too much work, normally cut down anyway

Positive News:
- Test driven development part of modern "agile" methods.

2 Systematic test case selection methods are already in place

Published in 1990 (Beizer)
Equivalence classes, boundary values, state transition testing even known since at least 1979.

Why don't we do it?
- Optimism: It probably works.
- Too much (boring) work
- We don't know the methods

News later:
- Test generation tools and possibility to run combination tests.

3 The tester shall evaluate, not show that it works

Destructive thinking.
Testing shall cover typical and non-typical use.
Trustworthiness grows from a thorough test where everything works well.

The usual scientific method.

BUT:

"We are so glad it works at all..."
And - Is the test trustworthy?

4 Test coverage should be measured

- Manually: Follow up of functions, transactions, data elements, test techniques used
  - Requires a lot of work
- Automatically: White box test using instrumentation tools
  - Tools exist for most languages
- Positive effect well documented in leading companies (HP, 1992)

"This is for safety critical software only"

5 Inspections are necessary

Documents always contain defects.
Reviews can be organized formally to increase their effectiveness and efficiency.
Inspections contain measurements and process improvement.

Known since 1976 at least.
Very much used in open source development.

Anybody who loves inspections?

6 Testing should be automated

Manual test is boring and unreliable.
Manual test is too expensive.
Retest and regression test is important.
Platform test is already a problem.

Manually testing everything is misusing people.

BUT: Exploratory testing!

Today: We HAVE the tools and the test design patterns (TestFrame, keyword driven test)

Regressions

The best experience reports show 1 new defect pr. 6 corrections / changes.

These can be found by regression test.

Best with automatic (programmed) test!

Parallel versions very demanding (one changed, the other one not).

7 Before automation we need a systematic process

Automating chaos makes chaos faster.
Automating a test requires extra resources.

You don't get such resources when fighting to test at all.
Still, testing is not mentioned in many life cycle models.

New trend after 2000: Test driven development!

8 Developers should unit test!

Lack of unit test overloads later test levels with defects.

But it is tested later anyway...

Modern methods (agile, XP) have strict requirements to unit test.
Modern tools make it easy to generate test environments.

9 Test cost depends on the quality of the system going into test

Problem: The worse the programs, the more failures.

A failure in test costs extra: Analysis, isolation, documenting, new installation, retest, regression test.

Tester should require quality before testing.

Isn't quality what customers expect anyway. Why don't we require it?

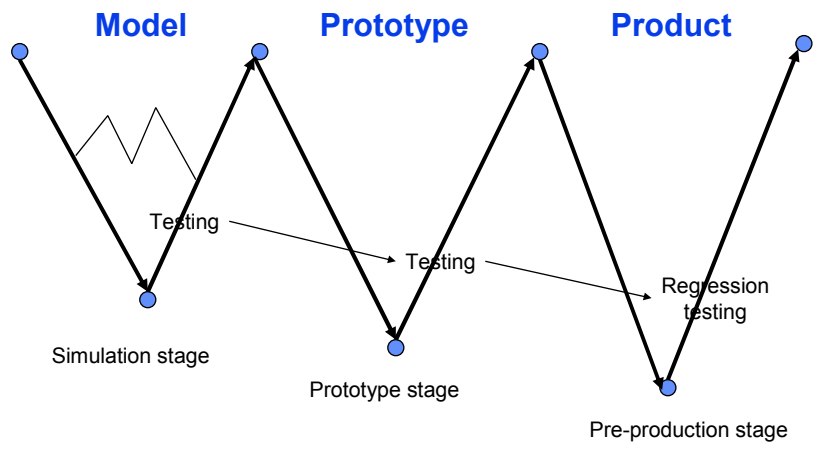
10 The V-model is useful.

- It requires test (at all).
- It requires early test preparation.
- It requires a test level for every development level, i.e. defect generation level.
- It must be adapted to incremental and evolutionary development: Much weight on regression test.
- It shows the place of testing, but it must not be misunderstood as a rigid model!

Yes, but we work in increments, or "agile"
(or without a plan at all)

George Box: All models are wrong, but some are useful.

The Modified V-model



Why don't we still do this?

Optimism about own work
Lack of education in software testing

But:
- **Tools are available.**
- **Testing is where the action is!**

Curricula are available:
- **Florida Institute of Technology Master degree in Software Testing,**
- **British Computer Society ISEB Certificate**
- **German ASQF Software Tester Certificate**
- **ISTQB Certification**
- **Standards for Safety Critical System Development**