

# Testing in Dynamic Environments

---

*Going beyond existing process models*

Prof. Walter Kriha, Hochschule der Medien  
Stuttgart,

Computer Science and Media Faculty

June 13, 2008



Overview

---

Test and Deployment  
A Look at Other Industries  
The Problem of Correctness

## 22 Years ago...

---

### Discoveries in development and testing

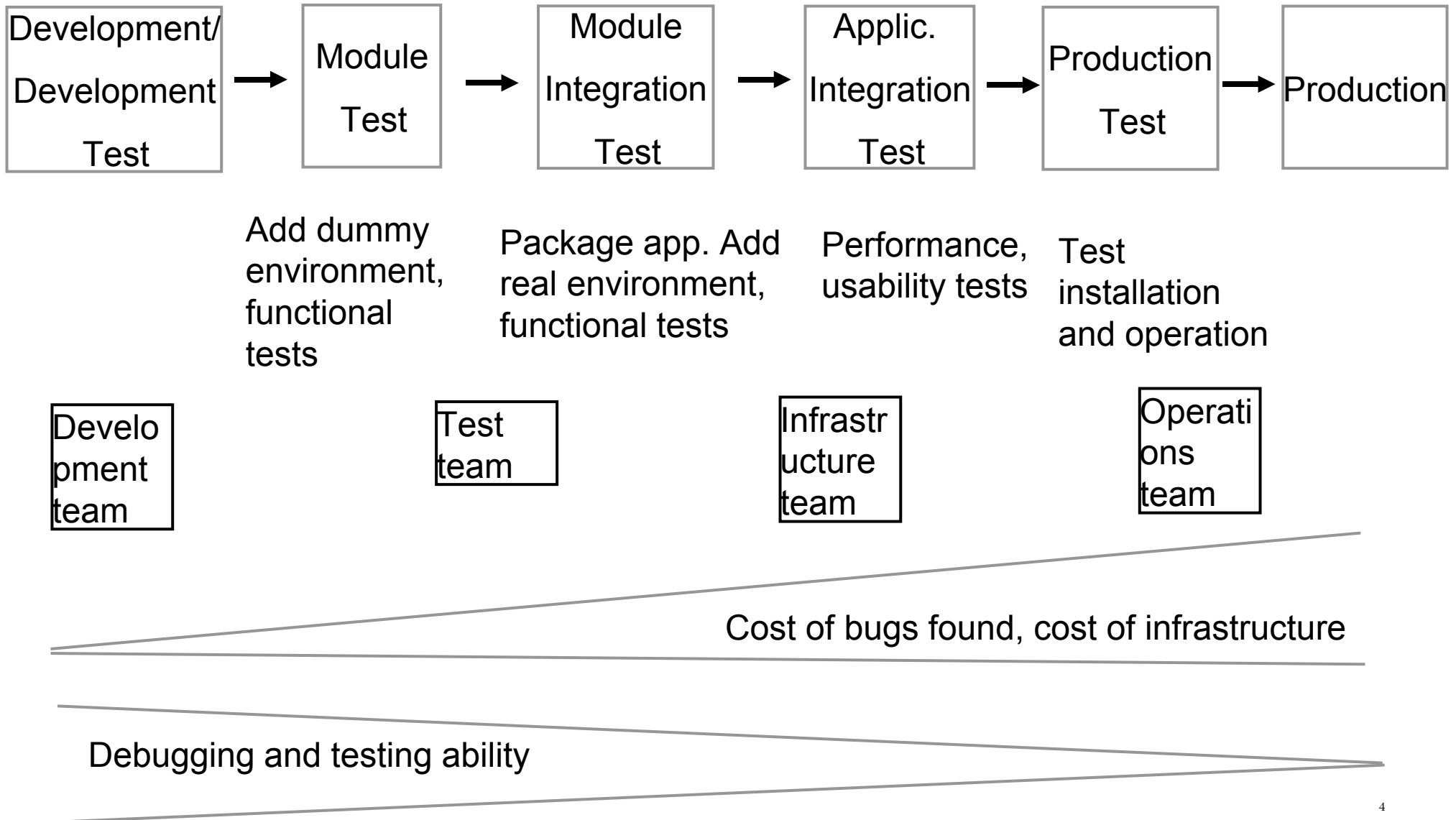
**22 years ago a young Unix system engineer without formal computer science education made some startling discoveries with respect to software quality and testing. At that time it was common knowledge that bugs found by the customer were the most expensive ones. Companies introduced test departments for final tests before deployment.**

**During an especially vexing problem hunt the developer and the test engineer became friends. The developer noticed several things:**

- 1. The closer the test engineer was involved during development the better the test software turned out to be**
- 2. The earlier the test engineer was involved in the design of APIs the better the APIs became (see: why APIs really matter)**
- 3. The number of bugs found later was dramatically reduced, as was the time spent for final tests.**
- 4. The software architecture and implementation was in some parts geared towards testability right from the start.**
- 5. Test and development became one activity but the two different departments had a problem with that approach**

# Today...

## A typical, high ceremony development, test and deployment process



# Problem Analysis

---

## How everybody suffers

The huge post-dev effort reduces release frequency

No time for automatic tests in case of fast changing software. This keeps rate of improvement low.

No or late feedback for developers. Bugs are not viewed as possible architectural problems

Agile teams do not profit from testing long after development. Agile methods in conflict with deployment process and organization.

The number of bugs found later in the chain is way too high. They are very hard to diagnose because of restricted developer access.

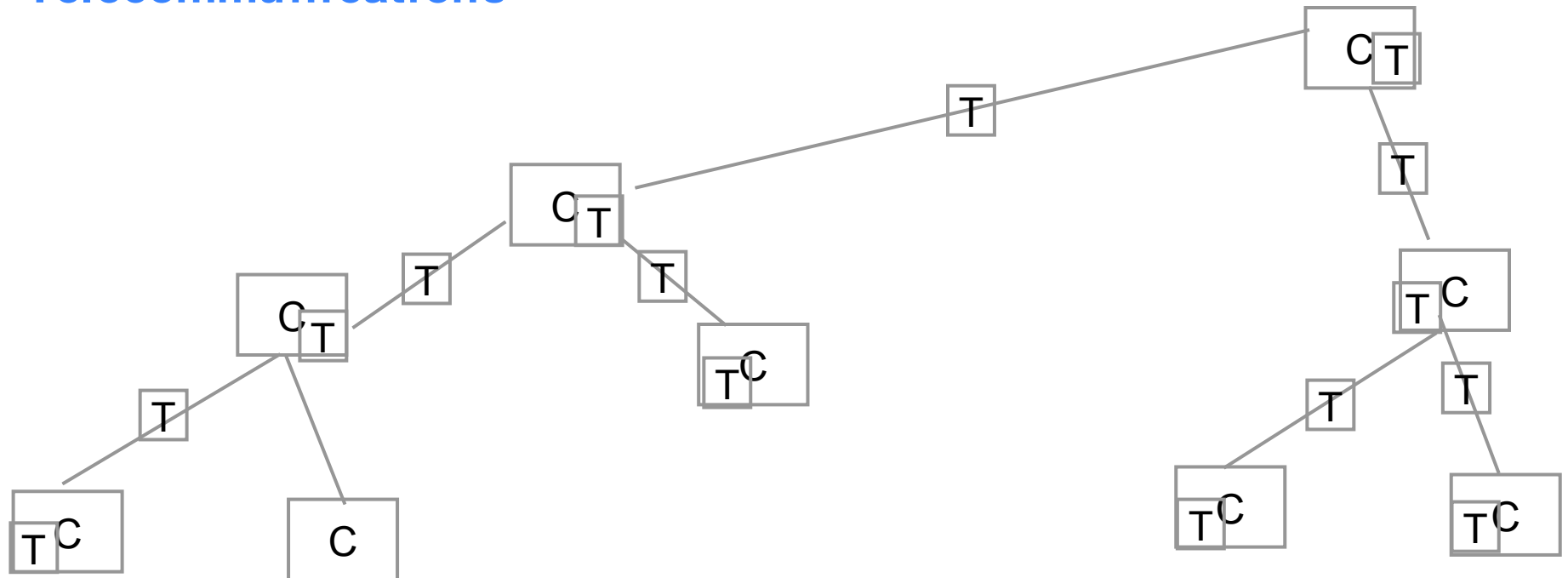
Several departments are involved which prohibits a common view on software problems and how to deal with them.

**How about combining development and test teams? How about creating a development environment that allows testing of all phases during development?**

# A look at other industries (1)

---

## Telecommunications

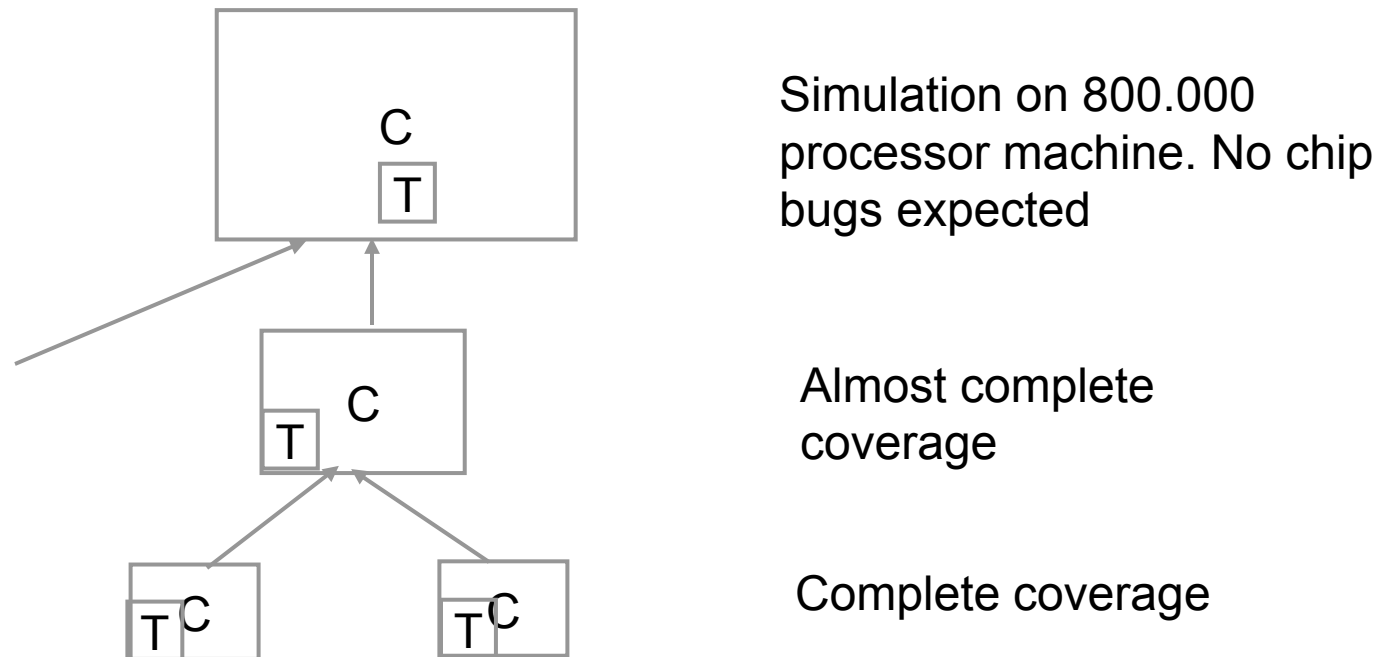


- Physical conditions and huge hierarchies are much more difficult than a simple 3-tier application
- Complete testing of boards and components
- Special test code within components that can be triggered at runtime
- Permanent monitoring of components and connections with event propagation

# A look at other industries (2)

---

## Chip Testing



- **Small numbers of transistors/gates are tested completely**
- **Small modules are only combined to larger ones after complete testing**
- **Design for testing: interfaces to test complete subsystems**
- **Test of complete chip really tests only the operating system, not the chip**
- **Nothing goes into production that has not been completely simulated (as long as computable)**

# Correctness

---

## Testing a highly customizable search engine platform

I wonder why I don't see document X in the hits?



Are these results really relevant?  
And for whom?

**Search engine testing created interesting problems: results are influenced by personal preferences, personal history, team history, department defaults, boosting and blocking by business specialists etc. What makes a result “correct”?**

# Future Testing (1)

---

## Architecture for Testing

Like logic languages software architecture will have to include an „explain“ feature that makes decisions tractable

Complex-Event Processing (CEP) causally explains inter-component activities

Dependency injection is understood as the core architectural pattern for virtualization – which in turn allows mock-ups of external services

Built in test-code allows testing of complete subsystems of an application. These tests are always available and cannot cause runtime problems. The differences between testing and monitoring are getting fuzzy.

# Future Testing (2)

---

## Organization for Testing

External test teams with or without scripting abilities will disappear.

Testing activities will be part of regular development work done by developers

Testing aspects will be part of architectural decisions right from the beginning of development

Functional test tools will be used by developers. They need to support agile and dynamic development. (see „Testing at Ascom“ talk by Christian Kaas in HDM video stream)

Logically separate aspects or concerns will no longer be split into different organizational parts.

# Resources

---

1. „Why API Design really matters“  
<http://www.kriha.de/krihaorg/blog6.html#i98>
2. Brian Marick, **An Alternative to Business-Facing TDD.**  
<http://www.exampler.com/blog/2008/03/23/an-alternative-to-business-facin>
3. Mike Bria, **Why Traditional Test-Automation Tools Stifle Agility**  
<http://www.infoq.com/news/2008/05/testobsessed-agile-auto-testing>
4. **Test and Quality Day at HDM, Video Stream of talks on agile, model-driven testing, industry approaches etc.,** [mms://stream.mi.hdm-stuttgart.de/testing\\_and\\_quality\\_day\\_ss08](mms://stream.mi.hdm-stuttgart.de/testing_and_quality_day_ss08)