

Software testing tools: from real-world software projects to educational contents

Klaus Bothe

*Institute of Informatics, Humboldt University – Berlin, Germany,
bothe@informatik.hu-berlin.de*

Workshop

**Cooperation at Academic Informatics Education across Balkan Countries and Beyond
Jelsa, Croatia, 2nd – 6th September 2019**

Contents

- ➔ ● Introduction
- Textbooks: What are the main issues of software testing?
- Other sources for testing issues
- Main issues and problems for testing in (our) software projects
- Which tools: TESTONA, ATOS, SOTA
- Summary

Our problem

Generally accepted:

- Software testing is a key discipline in software engineering
- Sources for educational contents:
textbooks, recommendations from authorities (e.g. IEEE)
- Software testing without tool support not successful in practice
- Thus, students should become familiar with some of them
– but which ones?

Which kind of testing tools should be used in educational environments based on main issues of software testing?

Our approach

Own experiences should be included.

- Evaluate (own) typical real-world software projects
- Main issues of our real-world software projects determine selection of testing tools

Our solution

There is no unique (best) software testing tool:
it depends on the activity to be supported

Main idea: testing tools for different needs:

- > Regression testing
- > Selection of test cases
- > Check the completeness of test cases

Contents

- Introduction
- ➔ ● Textbooks: What are the main issues of software testing?
- Other sources for testing issues
- Main issues and problems for testing in (our) software projects
- Which tools: TESTONA, ATOS, SOTA
- Summary

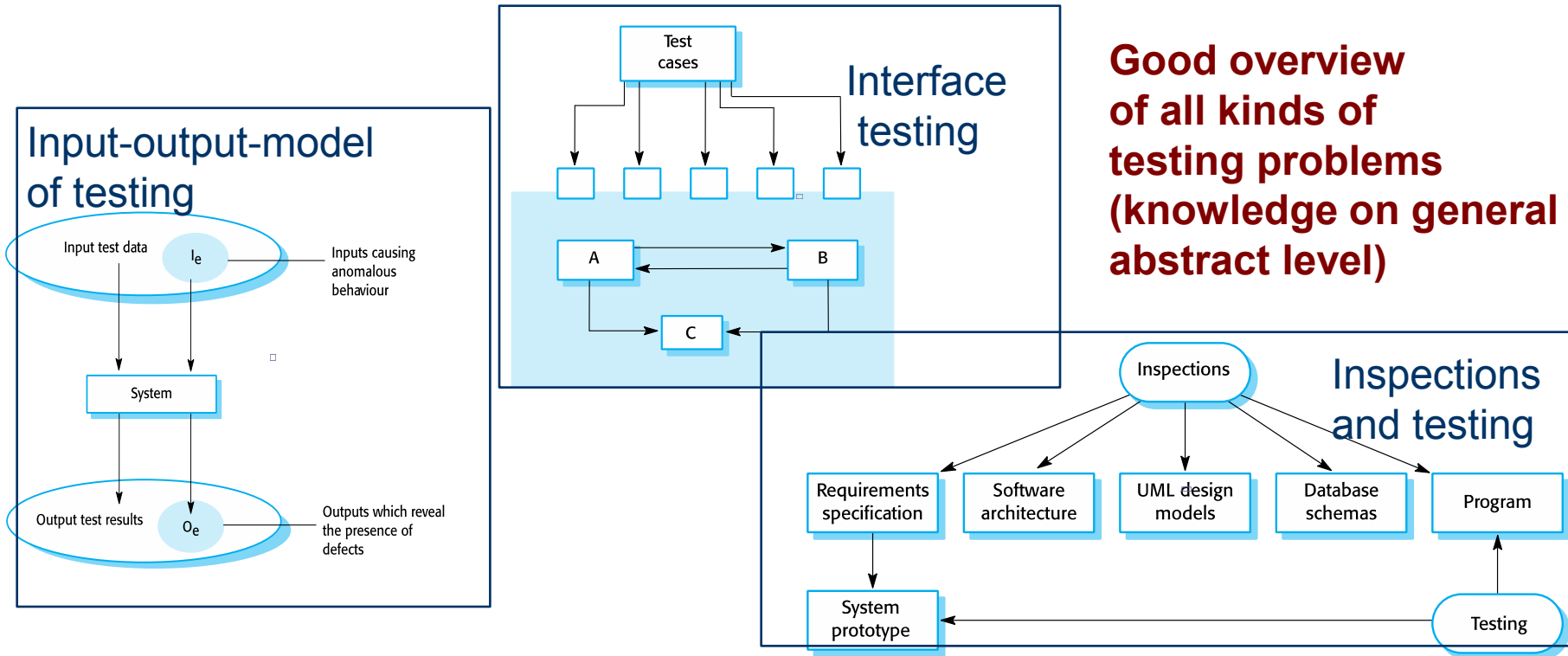


Textbooks

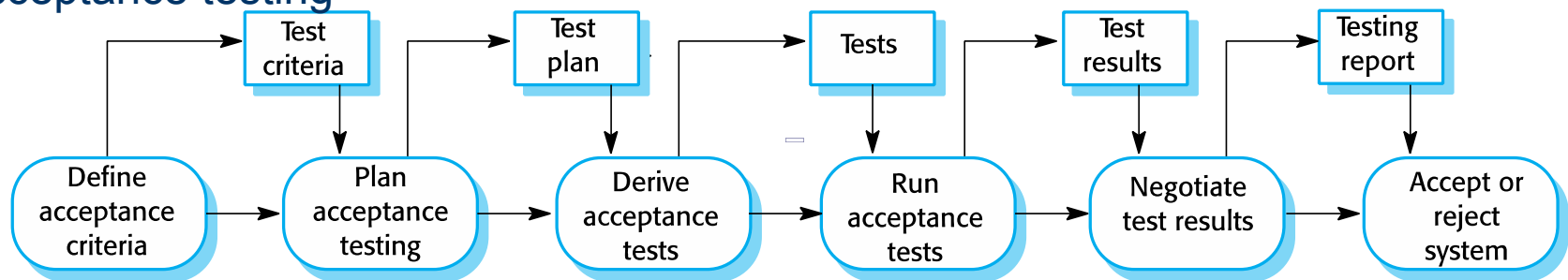
on general Software Engineering

Example: Sommerville, Software Engineering, 10th Edition, 2016

Educational contents in software testing: the case of Sommerville



Acceptance testing





Textbooks on Software Testing

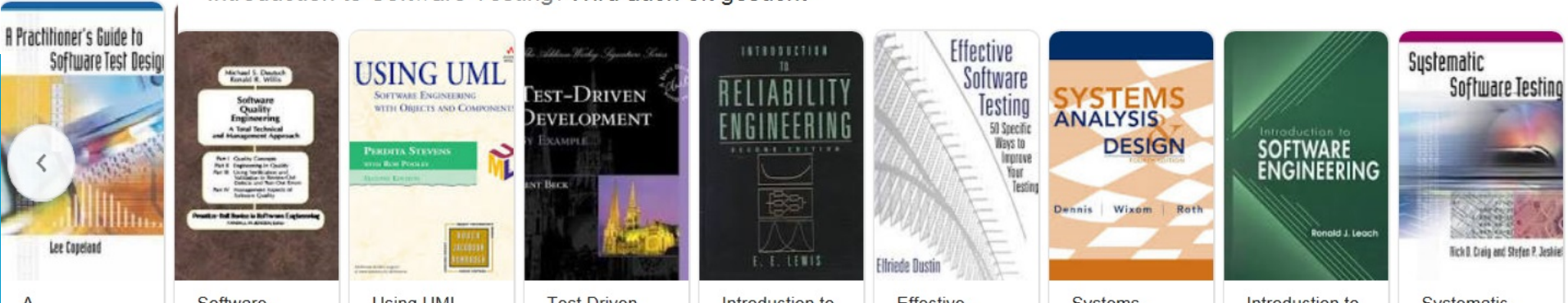
Introduction to Software Testing / Wird auch oft gesucht



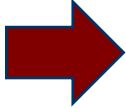
Introduction to Software Testing / Wird auch oft gesucht



Introduction to Software Testing / Wird auch oft gesucht



Contents

- Introduction
- Textbooks: What are the main issues of software testing?
-  ● Other sources for testing issues
- Main issues and problems for testing in (our) software projects
- Which tools: TESTONA, ATOS, SOTA
- Summary

SWEBOK: Software Engineering Body of Knowledge

(published by IEEE/ACM)

What belongs to the discipline of Software Engineering: A classified enumeration of fields

Guide to the SWEBOK®

Home | Contact us

 Français

 Español

Download the latest
Version (Feb. 16, 2005)

Also available in book
format

First International
Workshop
(Jul. 25-28, 2005)

Reviewer Demographics

Reviewer Response
Database Search Tool

Project Overview

Project Contributors

A Three-Phase
Approach

- Straw Man
- Stone Man
- Iron Man

Available Documents

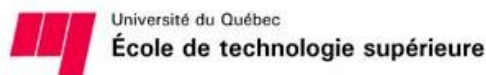
Guide to the Software Engineering Body of Knowledge

A project of the



Software Engineering Coordinating Committee

Project managed by



The following motion was unanimously adopted by the Industrial Advisory Board on February 6, 2004.

The Industrial Advisory Board finds that the Software Engineering Body of Knowledge project initiated in 1998 has been successfully completed; and endorses the 2004 Version of the Guide to the SWEBOK and commends it to the IEEE Computer Society Board of Governors for their approval.

Corporate support by



CANADIAN COUNCIL OF PROFESSIONAL ENGINEERS
CONSEIL CANADIEN DES INGÉNIEURS



SWEBOK 2014: Testing knowledge areas

Chapter 4: Software Testing

1. Software Testing Fundamentals
 - 1.1. Testing-Related Terminology
 - 1.2. Key Issues
 - 1.3. Relationship of Testing to Other Activities
2. Test Levels
 - 2.1. The Target of the Test
 - 2.2. Objectives of Testing
3. Test Techniques
 - 3.1. Based on the Software Engineer's Intuition and Experience
 - 3.2. Input Domain-Based Techniques
 - 3.3. Code-Based Techniques
 - 3.4. Fault-Based Techniques
 - 3.5. Usage-Based Techniques
 - 3.6. Model-Based Testing Techniques
 - 3.7. Techniques Based on the Nature of the Application
 - 3.8. Selecting and Combining Techniques

4. Test-Related Measures
 - 4.1. Evaluation of the Program Under Test
 - 4.2. Evaluation of the Tests Performed
5. Test Process
 - 5.1. Practical Considerations
 - 5.2. Test Activities
6. Software Testing Tools
 - 6.1. Testing Tool Support
 - 6.2. Categories of Tools

<http://www.swebok.org/>

ISTQB®: Certified tester



Search the site

FAQs | Contact us

- Home
- About us
- ISTQB® where you are
- Certification Path
- Exams
- Get Involved
- News
- Special Initiatives
- Downloads
- References

Certifying Software Testers Worldwide

ISTQB® has created the world's most successful scheme for certifying software testers.

As of September 2013, ISTQB® has issued [320,000 certifications](#) in over 100 countries [world-wide](#), with a growth rate of more than 12,000 certifications per quarter.

The scheme relies on a Body of Knowledge ([Syllabi](#) and [Glossary](#)) and [exam rules](#) that are applied consistently all over the world, with exams and supporting material being available in many languages.

Find the ISTQB® Board closest to you



EXPERT
CTEL

Test Management

Strategic Management
Operational Test Management
Managing the Test Team

Improving the Testing Process

Implementing Test Process Improvement
Assessing Test Processes

Test Automation

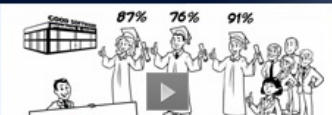
(planned for 2014)

Test Automation Engineering
Test Automation Management

Security Testing

(planned for 2015)

Smartshow



→ <http://www.istqb.org/>

→ ISTQB®:

International Software Testing Qualifications Board

→ Founded in 2002

→ Registered in Belgium

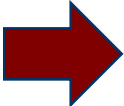
→ Until June 2017:

740.000 exams with 535.000 certifications in more than 120 countries



**A lot of material and
recommendations:
What to select?**

Contents

- Introduction
- Textbooks: What are the main issues of software testing?
- Other sources for testing issues
-  ● Main issues and problems for testing in (our) software projects
- Which tools: TESTONA, ATOS, SOTA
- Summary

(Own) Real-world software projects

Practical software projects (which are safety-critical)

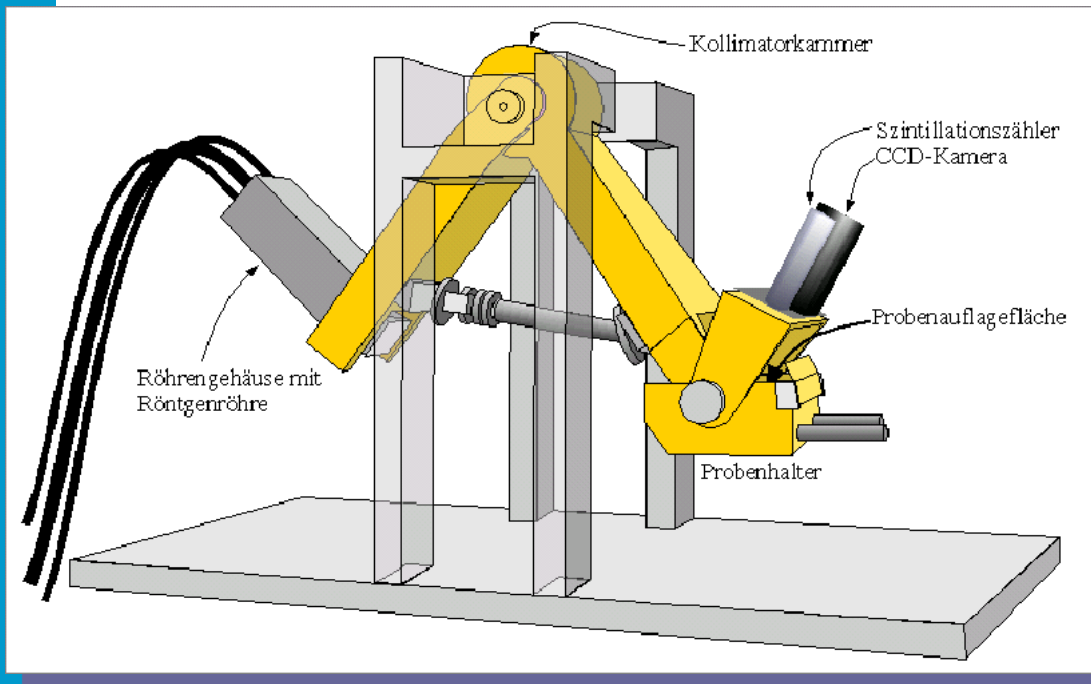
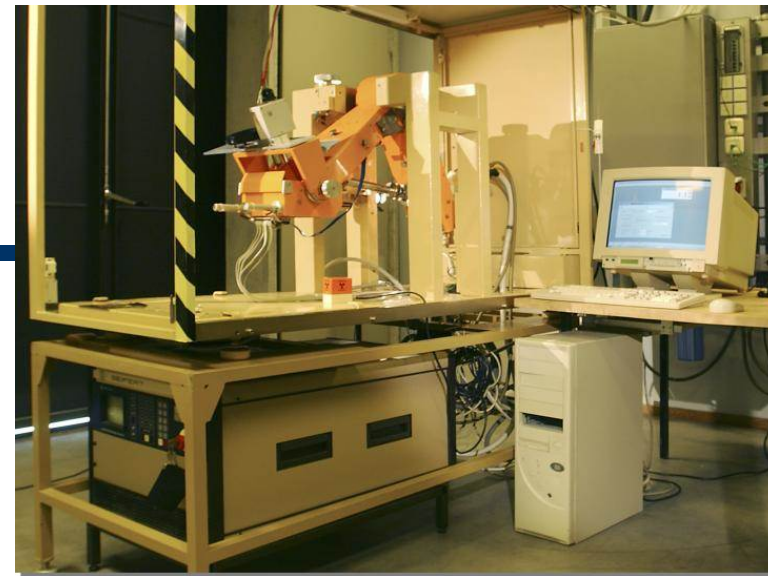
-> Bachelor and Master thesis with industry
(e.g. with car industry, suppliers to car industry)

-> ATEO: experiments in psychology

-> **XCTL**: control a device for experiments in physics
(safety-critical, long-term practical project, 2001-2014,
real customer: Institute of Physics, HU)

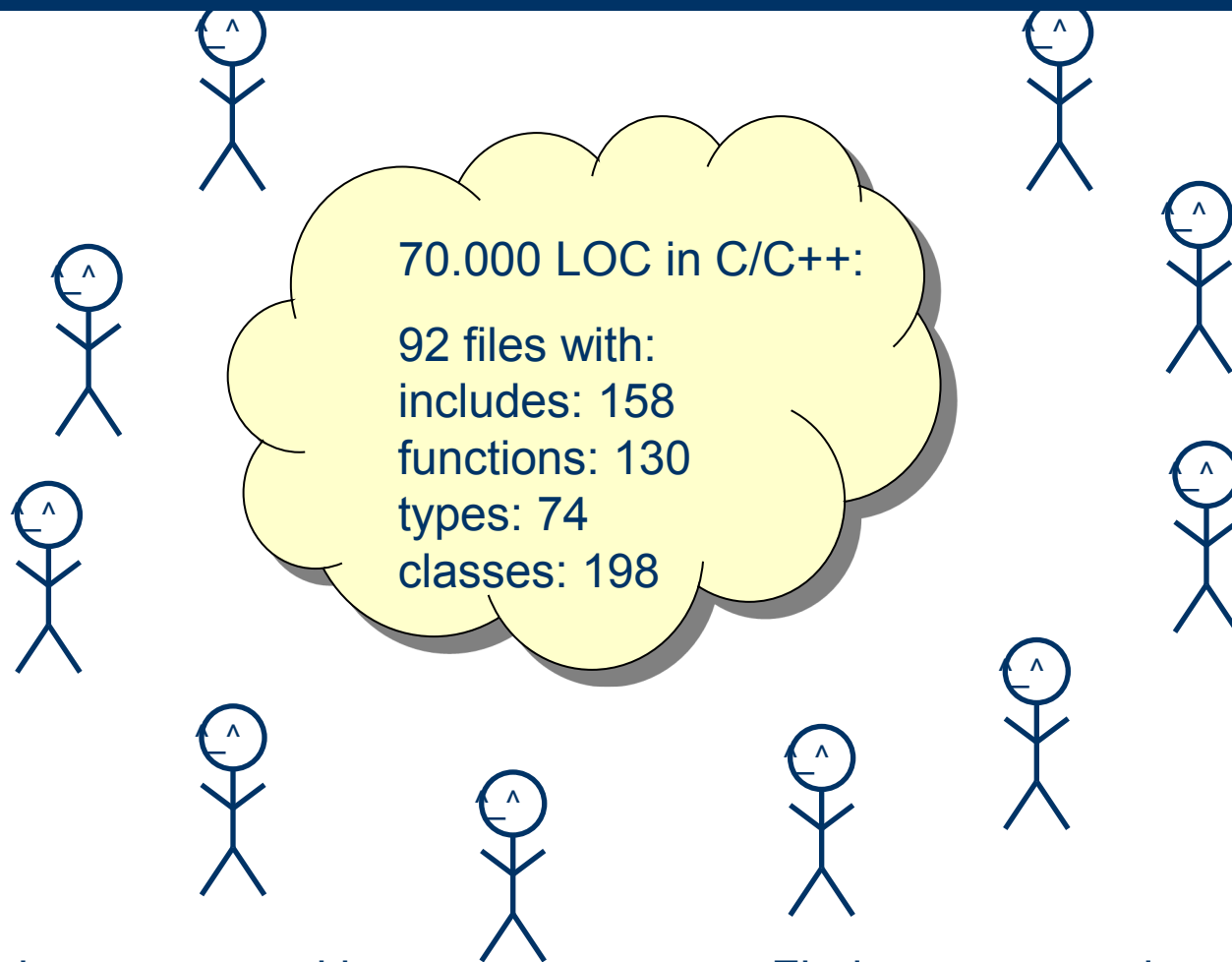
Real-life project XCTL (control a device for experiments in physics)

working
place



X-ray topography
camera

Real-life project XCTL (control a device for experiments in physics)



Many developers are working
at different parts of the system

Find errors as early as possible:
test very often (but: test takes 2 days)

Main issues of software testing in our projects (esp. XCTL)

- How to automate regression testing, i.e. to prove that there is no regression after modifications of software?
- How to support the development test cases?
- How to check the completeness of test cases?

Contents

- Introduction
- Textbooks: What are the main issues of software testing?
- Other sources for testing issues
- Main issues and problems for testing in (our) software projects
- ➔ ● Which tools: TESTONA, ATOS, SOTA
- Summary

Main issues of software testing in our projects

- How to automate regression testing, i.e. to prove that there is no regression after modifications of software?
- How to support the development test cases?
- How to check the completeness of test cases?

→ Each issue should be supported by a testing tool

Real-life project XCTL: three testing tools



70.000 LOC in C/C++:

92 files with:
includes: 158
functions: 130
types: 74
classes: 198

Testing tools for different needs:

TESTONA:

Find all necessary test cases

SOTA:

Check the quality of test cases:
measure coverage degrees

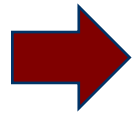
ATOS:

Automatic regression testing
(repeat the test after modification)

Many developers are working
at different parts of the system

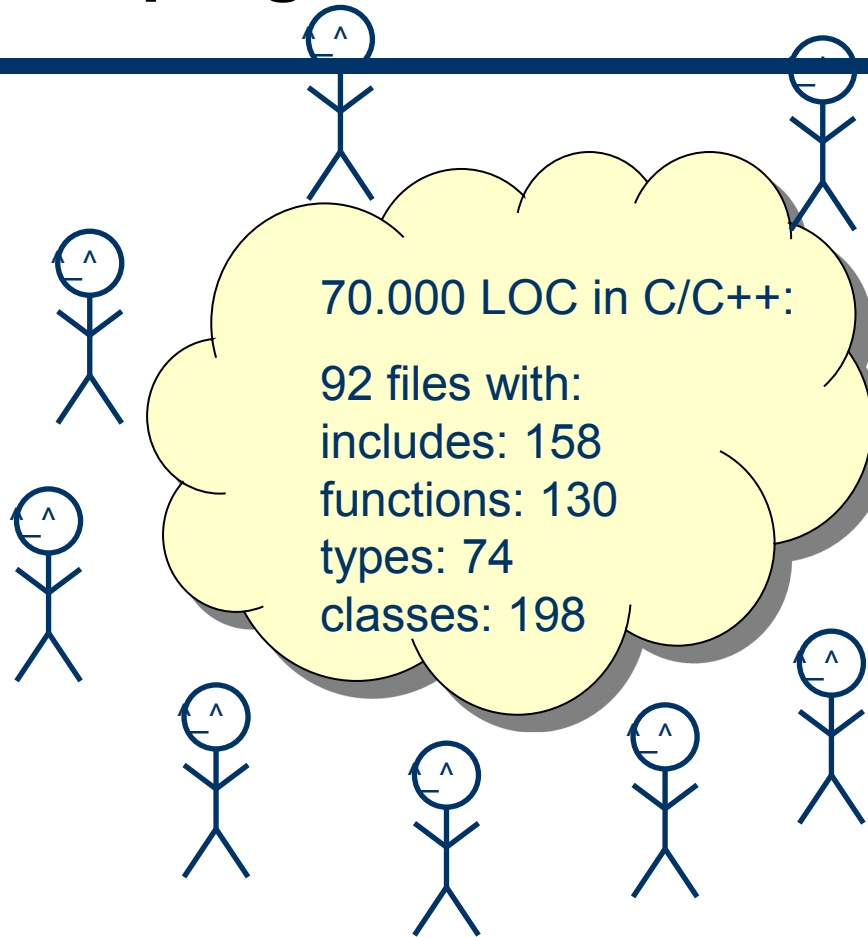
Find errors as early as possible:
test very often (but: test takes 2 days)

Main issues of software testing in our projects



- How to automate regression testing, i.e. to prove that there is no regression after modifications of software?
- How to support the development test cases?
- How to check the completeness of test cases?

Problem: systematic testing after each program modification



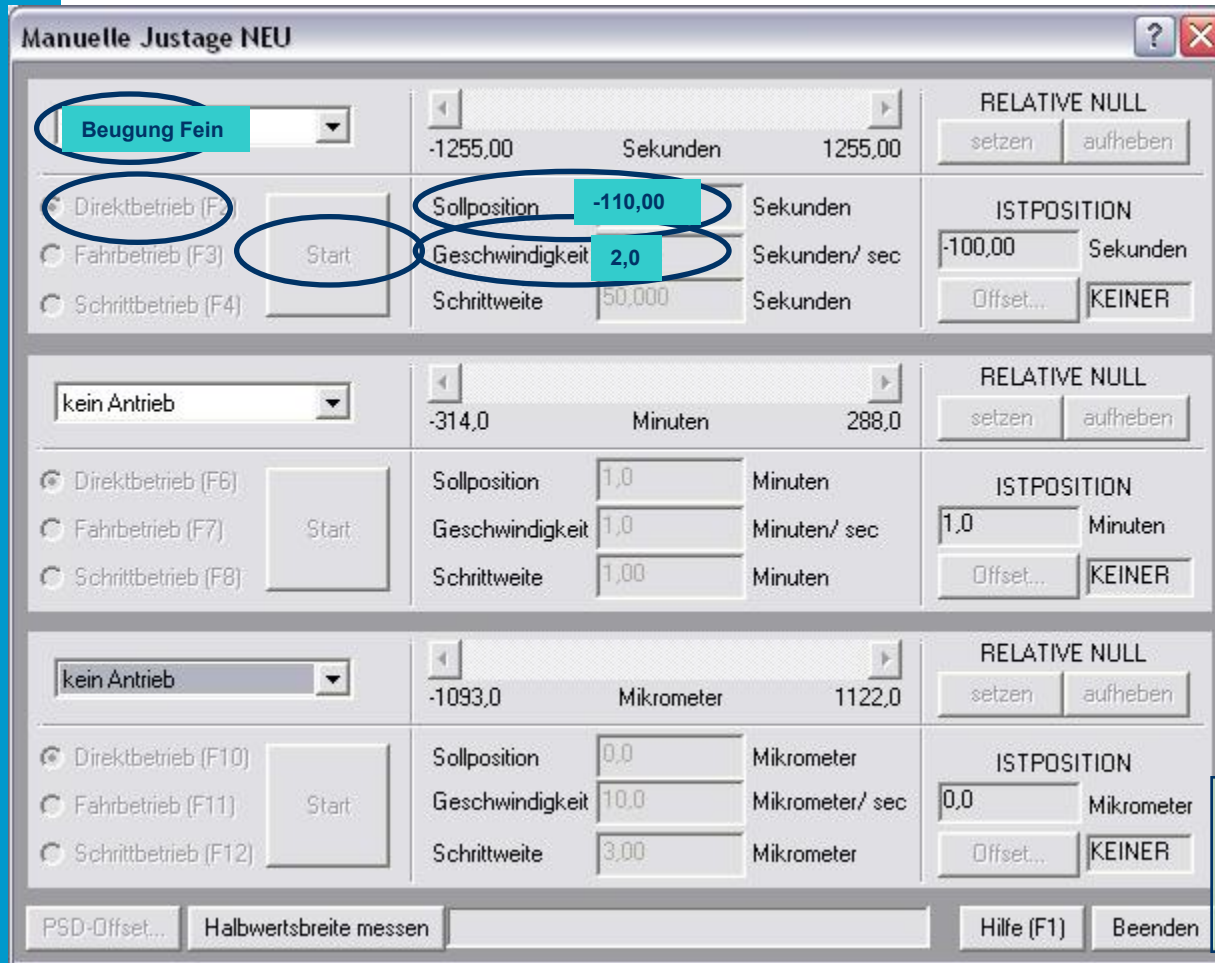
Regression testing

- Regression testing:
- Testing that after modifications there are no new errors included (no step backward = no regression)
 - The same test cases before and after modifications
 - Tool support:
Automatic run and documentation

Many developers are working at different parts of the system

**Instead of manual testing 2 days:
now only 1 hour**

GUI-oriented automatic regression testing: sequence of test activities

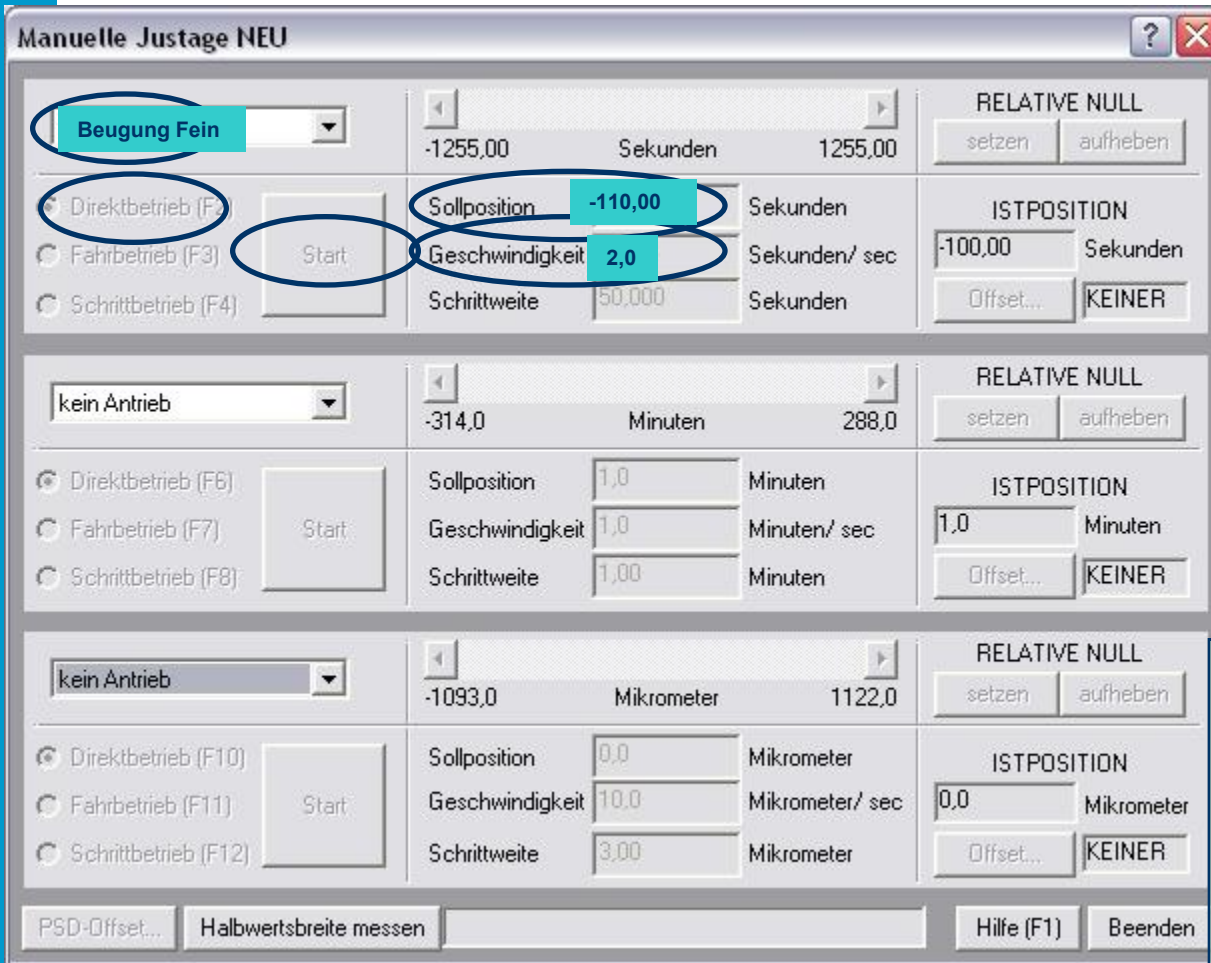


Test sequence:

- Start the system XCTL
- Open window 'Manuelle Justage'
- Initial state: of dialogue box; source: .ini-File
- Activities on the dialogue box
 - Combo box: select motor (Beugung Fein)
 - Radio button: Direktbetrieb
 - Edit box: Sollposition: -110,00
 - Edit box: Geschwindigkeit: 2,0
 - Button: Start
- Close dialogue box 'Manuelle Justage'
- Stop the system XCTL

**Manual input into the GUI:
Time-consuming, error-prone
- If you repeat it again and again**

GUI-oriented automatic regression testing: Capture and replay tool ATOS for GUI systems



Test sequence:

- Start the system XCTL
- Open window 'Manuelle Justage'
- Initial state: of dialogue box; source: .ini-File
- Activities on the dialogue box
 - Combo box: select motor (Beugung Fein)
 - Radio button: Direktbetrieb
 - Edit box: Sollposition: -110,00
 - Edit box: Geschwindigkeit: 2,0
 - Button: Start
- Close dialogue box 'Manuelle Justage'
- Stop the system XCTL

• Capture:

Manual input into the GUI:
The test tool ATOS stores the input in a test script file.

• Replay:

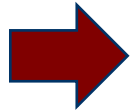
Later on, under the control of ATOS, this script file directs the run of the system (regression testing)

ATOS

WinRunner

Main issues of software testing in our projects

- How to automate regression testing, i.e. to prove that there is no regression after modifications of software?



- How to support the development of test cases?
- How to check the completeness of test cases?

TESTONA: Classification Tree Method

Developed by Daimler-Chrysler:

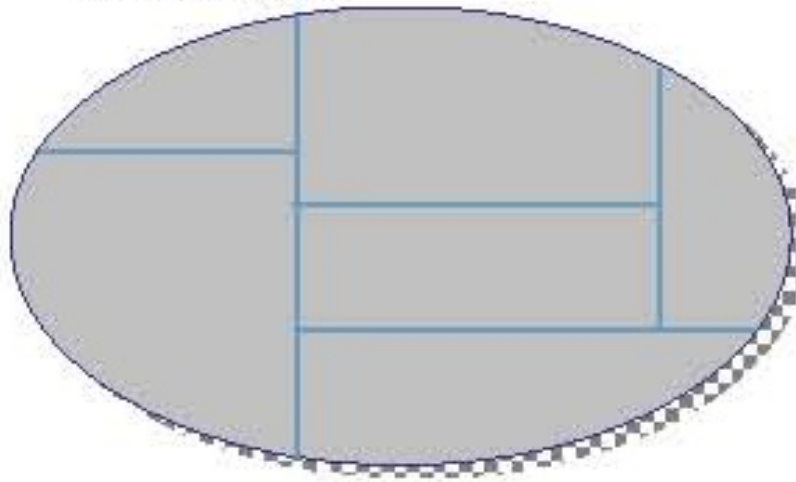
- **Originally: Embedded into a test system for car industry**

<http://www.testona.net/en/index.html>

- **Widely propagated usage:
Airbus, Mercedes, BMW, Audi etc.**
- **Free academic version**

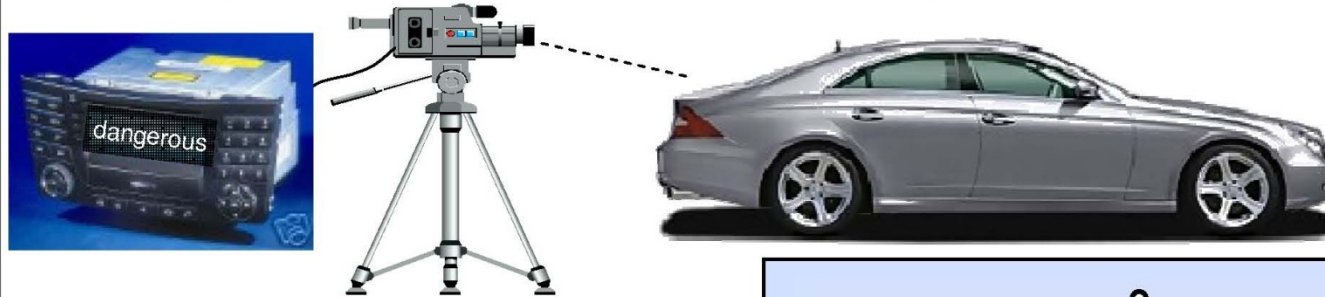
Classification Tree Method

Basic principle:
Classification of
input data space

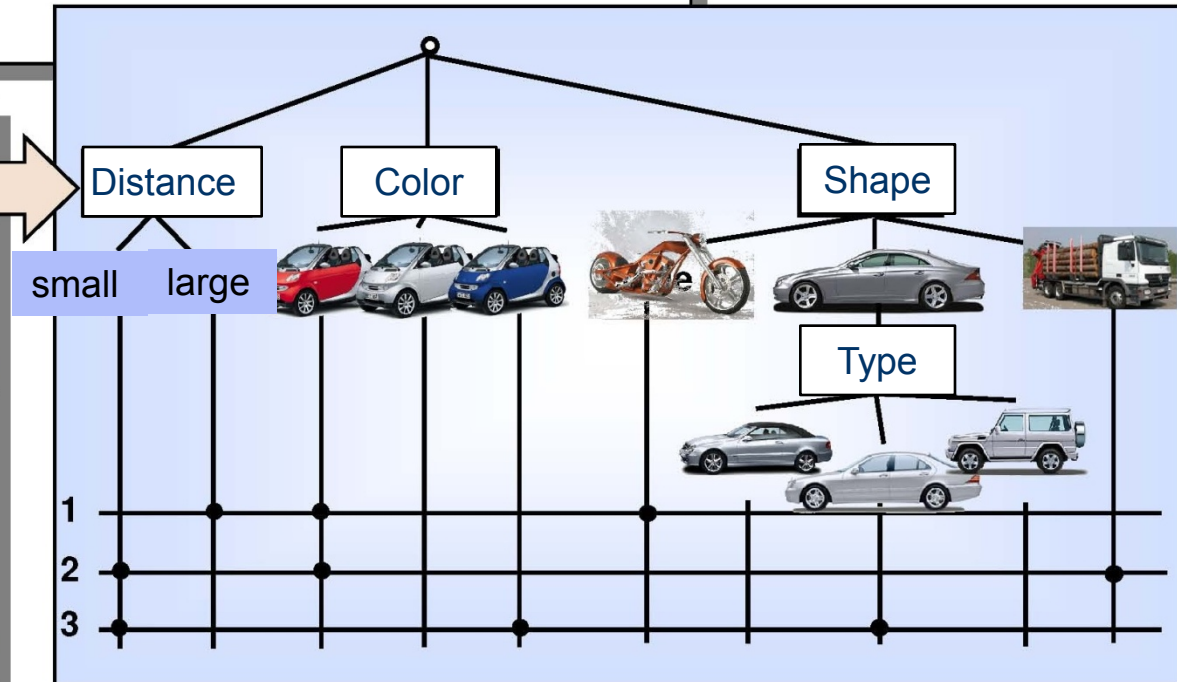
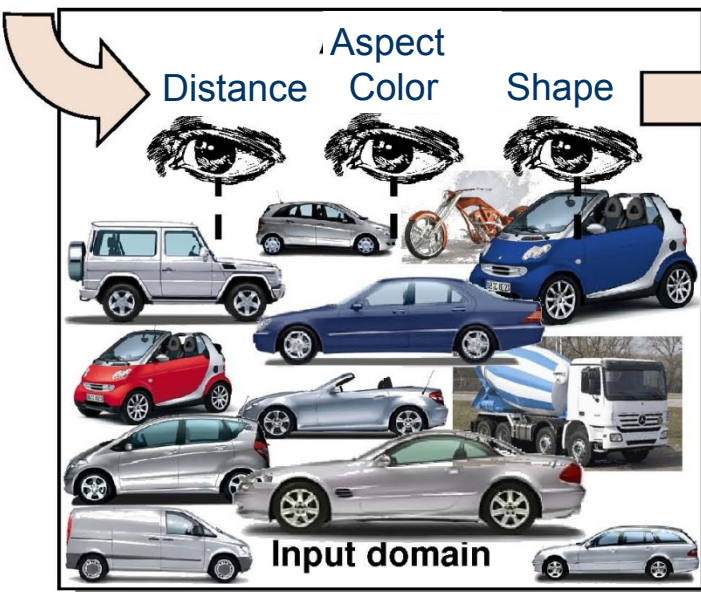


Driving assistant: Determine the distance between objects

Exemplary test object: computer-vision system, distance to the preceding vehicle determined



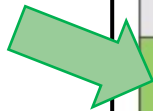
Autonomous car driving



Source of case-study: Wegener, DaimlerChrysler

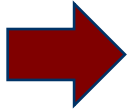
TESTONA Editions

TESTONA Light	TESTONA Express	TESTONA Professional	TESTONA Enterprise
Free Edition with reduced functionality	Edition for the cost-effective entry into professional software testing	Professional Edition conceived for sophisticated software testing	Enterprise-Edition with full scope of performance and all features tailored to individual needs
AddOns not available	AddOns not available	Alle AddOns available	All AddOns included (e.g. AUTOSAR, Matlab)
Support & Maintenance not available	Support & Maintenance available	Support & Maintenance package available (incl. project support and more)	Incl. 2-days On-site Training
Only Node Locked Licence	Only Node Locked Licence	Node Locked, Hard Locked, Floating-Licence	Floating-Licence incl. Borrow-Function
Details about Features + Features	Details about Features + Features	Details about Features + Features	Details about Features + Features
		Additional features included such as Requirements tracing, colouring	
		Numerous Import/Export-connections (e.g. DOORS, QualityCenter)	
Free Download	Buy TESTONA Express	Buy TESTONA Professional	Buy TESTONA Enterprise



Main issues of software testing in our projects

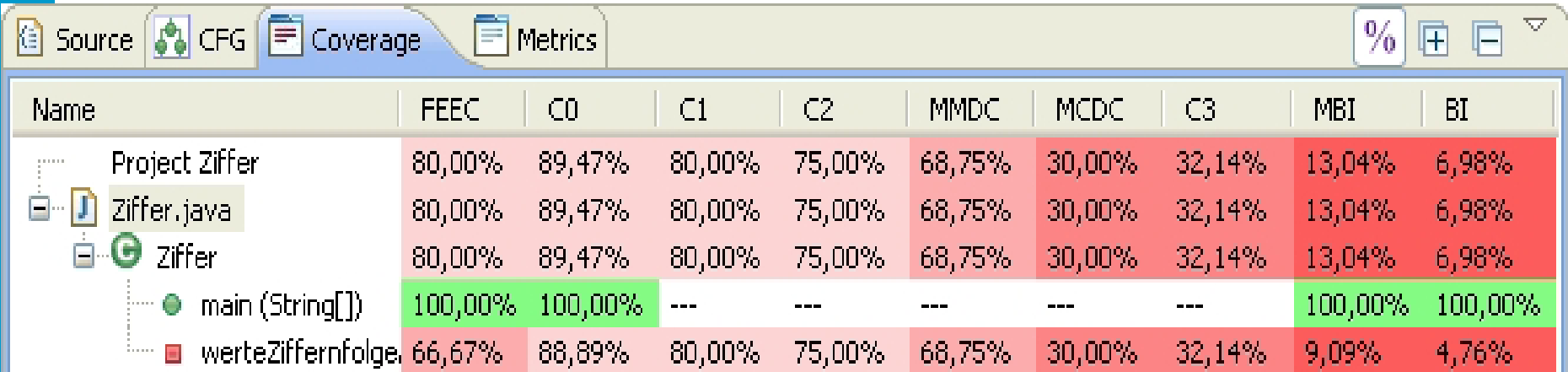
- How to automate regression testing, i.e. to prove that there is no regression after modifications of software?
- How to support the development test cases?
- How to check the completeness of test cases?



SOTA ...

- **Delivers coverage degrees**
for a given test data set and a program
(based on *structure-oriented testing / white box testing*)
 - **Visualization of coverage**
- **Quality of test data assessed**

SOTA: coverage degrees for nine criteria



The screenshot shows the Coverage tool window in an IDE. The window has tabs for 'Source', 'CFG', 'Coverage', and 'Metrics'. The 'Coverage' tab is active, displaying a table of coverage metrics for the project 'Ziffer'. The table has columns for 'Name', 'FEEC', 'C0', 'C1', 'C2', 'MMDC', 'MCDC', 'C3', 'MBI', and 'BI'. The rows represent different levels of the project hierarchy: 'Project Ziffer', 'Ziffer.java', 'Ziffer', 'main (String[])', and 'werteZiffernfolge'. The 'main (String[])' row is highlighted in green, indicating 100% coverage for all metrics. The other rows are highlighted in red, indicating lower coverage percentages.

Name	FEEC	C0	C1	C2	MMDC	MCDC	C3	MBI	BI
Project Ziffer	80,00%	89,47%	80,00%	75,00%	68,75%	30,00%	32,14%	13,04%	6,98%
Ziffer.java	80,00%	89,47%	80,00%	75,00%	68,75%	30,00%	32,14%	13,04%	6,98%
Ziffer	80,00%	89,47%	80,00%	75,00%	68,75%	30,00%	32,14%	13,04%	6,98%
main (String[])	100,00%	100,00%	---	---	---	---	---	100,00%	100,00%
werteZiffernfolge	66,67%	88,89%	80,00%	75,00%	68,75%	30,00%	32,14%	9,09%	4,76%

FEEC: Function-Input-Output-coverage

C0: Statement coverage

C1: Branch coverage

C2: Simple condition coverage

MMDC: Minimal multiple conditions coverage

MCDC: Modified Condition Decision Coverage

C3: Multiple conditions coverage

MBI: Modified Boundary-Interior path coverage

BI: Boundary-Interior-Path Test

Control-flow graph and coverage visualization

The screenshot displays the SOTA - Project Ziffer IDE interface. The main window shows a control-flow graph (CFG) for the method `werteZiffernfolgeAus (String)`. The graph consists of nodes representing code blocks, connected by edges with associated coverage counts. The nodes are:

- `werteZiffernfolgeAus (String)` (green)
- `iteration` (green)
- `iter-body` (green)
- `if` (green)
- `true` (green)
- `false` (green)
- `if` (yellow)
- `true` (red)
- `false` (green)
- `return` (green)
- `return` (red)

The coverage counts on the edges are: 3 (entry to iteration), 6 (iteration to iter-body), 3 (iteration to iter-end), 6 (iter-body to if), 4 (if to true), 2 (if to false), 3 (if to return), 0 (if to true), 2 (true to if), 2 (false to if), 2 (if to return), 0 (if to true), 3 (return to exit), and 0 (return to exit).

The TestLogs panel on the left shows the following test cases:

- test ..
- test .2
- test 1
- test 1.1
- test c
- test empty

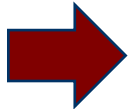
The source code at the bottom of the window is:

```
private static double werteZiffernfolgeAus(String inZiffernString) {  
    double wert = 0.0;  
    ...  
}
```

The status of the source code is CLEAN, and the status of the parsed code is CLEAN.

Contents

- Introduction
- Textbooks: What are the main issues of software testing?
- Other sources for testing issues
- Main issues and problems for testing in (our) software projects
- Which tools: TESTONA, ATOS, SOTA
- Summary



Conclusion

There is no unique (best) software testing tool:
it depends on the activity to be supported

Selection of three different kinds of testing tools
for different purposes:

- ATOS/ATOSj *) : Regression testing for GUI software
- TESTONA +): Systematic selection of test cases
- SOTA *): Check the completeness of test cases
with respect to different criteria of structure-oriented testing
(white box testing, e.g. branch coverage)

*) own development

+) industry product (Daimler, free academic version)

→ Students worked with three different tools in practical assignments

Own free software tools



Prof. Dr. K. Bothe

Overview

SOTA

ATOSj

Software Engineering Group Testing Tools: SOTA

SOTA

SOTA is a tool for **static program analysis and structure-oriented program testing**. The static program analysis provides several software metrics such as cyclomatic or essential complexity as well as the control flow graph of each method. Moreover it provides several code coverage measures like C0,C1,C2,C3, MMCC and MCDC. These measures are determined by instrumenting of the original source code and the logging of the control and data flow during the test execution. SOTA provides multiple ways of usage: the manual program test, in conjunction with other external test systems (such as ATOSj) or even integrated in an automatic test system.

ATOSj

ATOSj is a tool for the development and execution of **automated regression tests of Graphic User Interfaces** (GUIs) of Java applications, whose GUIs have been developed using the Swing or the SWT framework. ATOSj provides a graphical user interface for the creation, (automatic) execution and evaluation of HTS test sequences. The test sequences can be created and executed using the *capture-and-replay*-technique. ATOSj provides also support for editing captured test sequences or for creating them from scratch. Moreover ATOSj also supports the export of the test results as PDF reports.



Thank you