

12. Validation Tools

Lecturer: Dr. Sebastian Götz

Prof. Dr. rer. nat. Uwe Aßmann

Institut für Software- und Multimediatechnik

Lehrstuhl Softwaretechnologie

Fakultät für Informatik

TU Dresden

<http://st.inf.tu-dresden.de>

Wintersemester 2017, 15.11.2017

- 1) Code Analysis Tools
 - 1) Programmanalyse
- 2) Code-Centric Test Environments
 - 1) Coverage-based test tools
 - 2) JouleUnit energy test framework
 - 3) Eclipse-based test tools
- 3) Requirements-Driven Test Environments
 - 1) Classification tree method and Tessy
- 4) Model-Driven Test Environments
 - 1) MATE
 - 2) Andere

For metrics, control-flow and value-flow analysis

12.1 CODE ANALYSIS TOOLS

Functionality of Kalimetrix Logiscope

- ▶ <http://www.kalimetrix.com/home/221-migrate-from-logiscope-ibm-to-logiscope-2012>
- ▶ Different vendors: Telelogic North America Inc., Irvine, USA (Hersteller des Requirement Management Systems DOORS), Rational, IBM, since 2012 Kalimetrix
- ▶ <http://www.kalimetrix.com/en/logiscope>
 - ▶ Windows, Linux
 - ▶ Support for languages C, C++, Java, Ada, C#, Visual Basic.
- ▶ Coverage testing
- ▶ Test delivers
 - Trace protocols
 - Untested branches
 - Visualization of control flow graphs and call graphs
- Computation of metrics
- ▶ Test definition:
 - Instrumentation of code
 - Graphic analysis of test results
 - Generation of test documentation

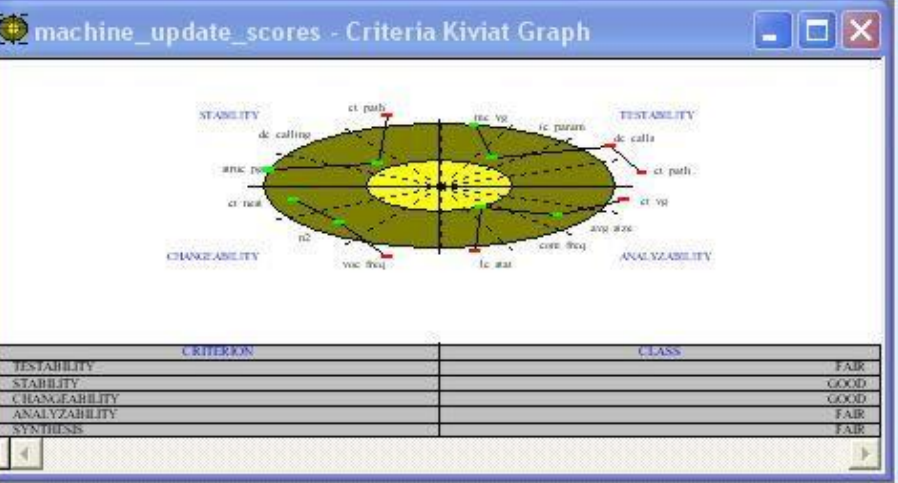
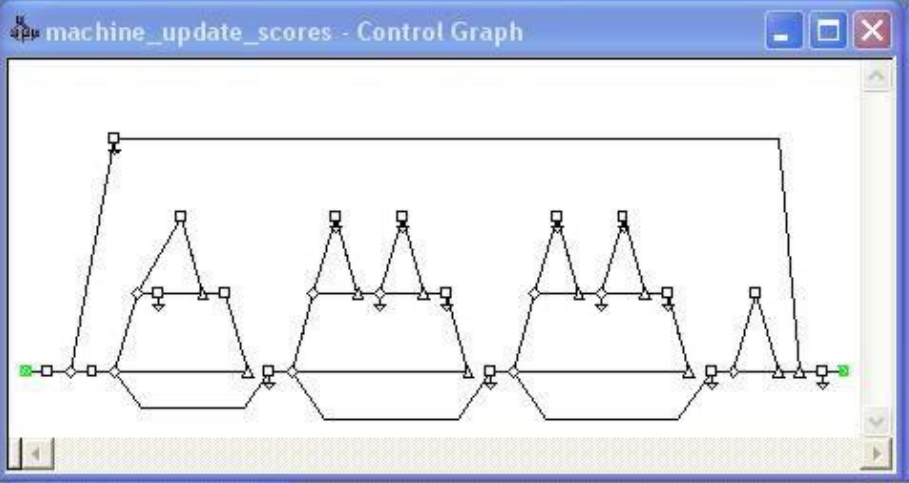
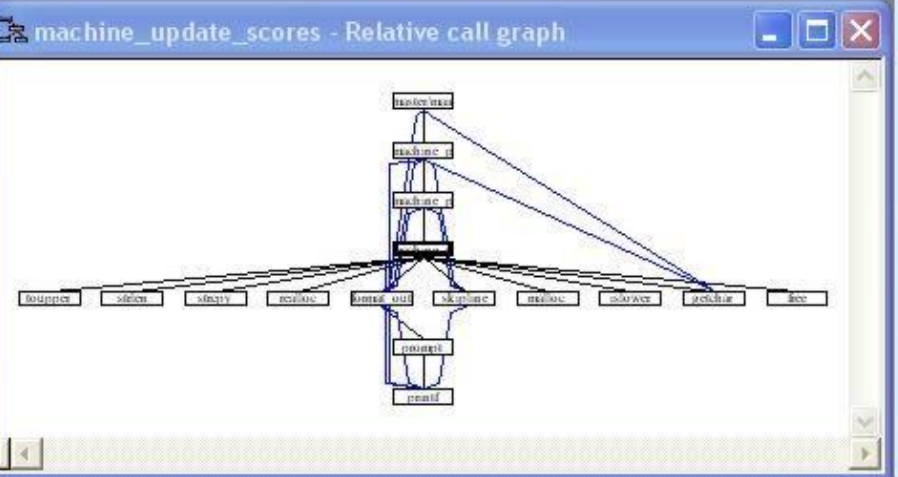


- Workspace1 - Call gra
- consistent
 - end_game
 - find_digit
 - format_output
 - get_code_mac
 - get_code_play
 - help
 - hi_scores_disp
 - hi_scores_writ
 - instruction
 - machine_plays
 - machine_print
 - machine_read
 - machine_upde
 - make_code
 - master/main
 - play
 - player_plays
 - player_score
 - print_help
 - print_instructio
 - print_instructio
 - print_score_pk
 - prompt
 - refresh
 - rest
 - score_mac
 - score_player
 - set_dummy

```

103     }
104     number [i] = car;
105     car = getc (hi_scores_file);
106     i++;
107 }
108
109     number [i] = '\0';
110     (hi_scores_tab [last_hi_score]).score =
111     free (number);
112
113     car = getc (hi_scores_file);
114     last_hi_score++;
115 }
116

```



Works...

Loading CodeReducer data...
 Loading RuleChecker data...
 + loaded 55 function(s)
 + loaded 0 class(es)

<http://www.kalimetrix.com/en/logiscope/qualitychecker>

- Quality Checker: definition and checking of software metrics
 - Quality characteristics
- Rule Checker:
 - Definition of programming guidelines and rules in Tcl or Perl
 - Check the rules of standards for regulation (MISRA)
- Test Checker: Coverage checker
 - Check acceptance of diverse standards such as IEC 61508-3, DO178B (Arial)
 - Generation of test reports
- Code Reducer:
 - Code clone detection
- Viewer: Visualizer for graphs

Sonargraph Metrics Tool

- Hello2Morrow <http://www.hello2morrow.com>
 - <http://www.hello2morrow.com/products/sonargraph>
- Static and metrics-based analyses for the ASG:
 - Architectural analysis (package dependencies, layering,...)
 - Monitoring the compliance to architectural rules
 - Finding “bad smells”, opportunities for refactoring
- Trend computations for metrics
- 3D visualisation of results
- Specification of architecture in an architectural DSL

12.2 CODE-CENTRIC TEST- FRAMEWORKS

12.2.1 CODE COVERAGE TOOLS



Control-Flow Oriented White-Box Test (Code Coverage)

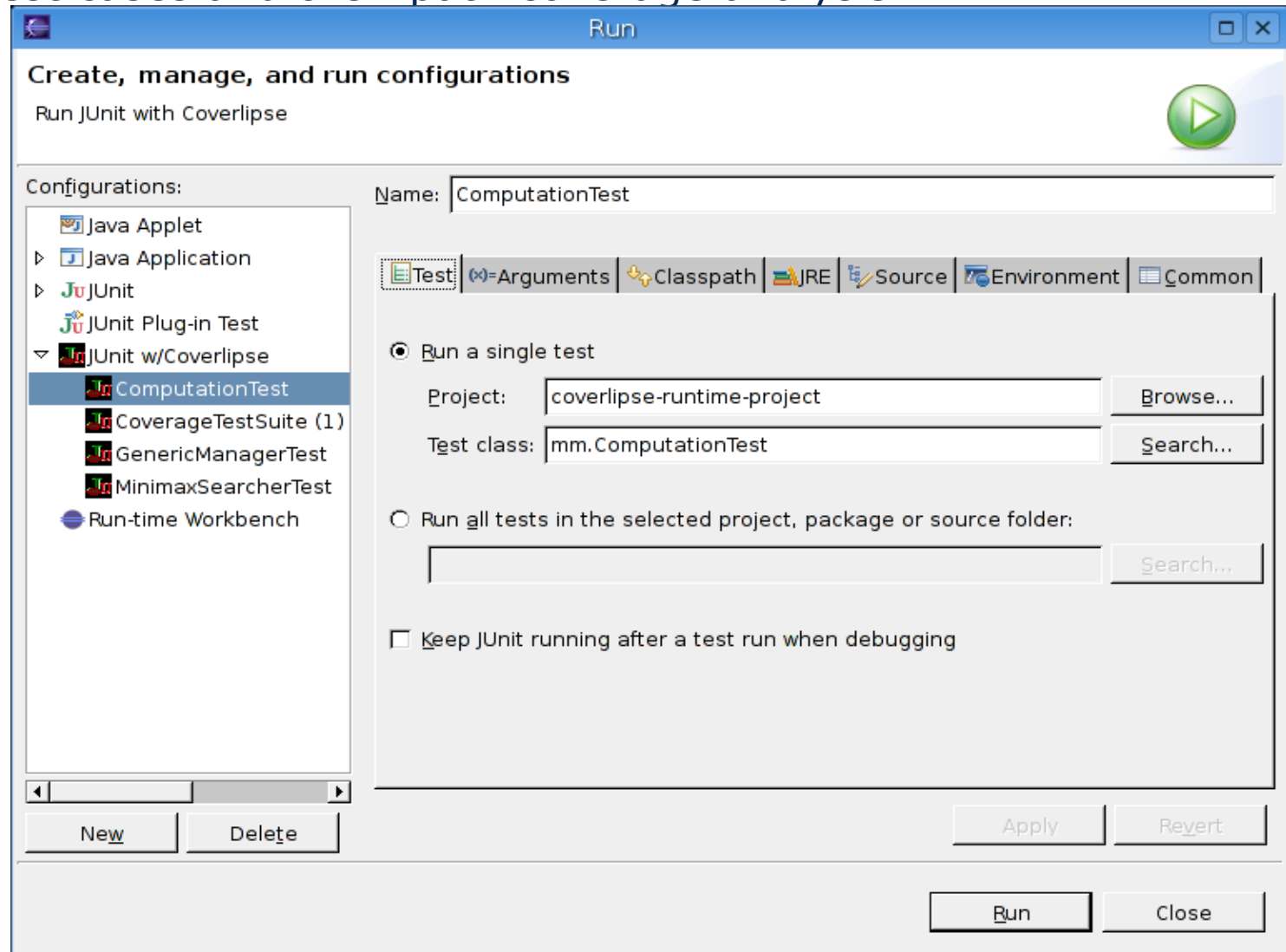
Coverage class	Technique	Purpose
Statement	How many statements are covered by how many test cases?	Discover dead code
Condition (Alternative)	How many branches are covered by how many test cases?	Cover all edges in control-flow graph
Combination of conditions	How many combinations of subsequent branches are covered by how many test cases? Coverage of acyclic paths	Problem: combinatoric explosion
Combination of independent conditions	All combinations of those conditions influencing the result of the program independently	Reduction of the effort
Path	Coverage also of cyclic paths	Im Allgemeinen unmöglich; Einschränkung auf Durchlaufsschranke k
Boundary Test	Coverage of all paths, with at most 1 run through a loop	Loop bound is $k \leq 1$
Interior Test	Coverage of all paths, with at most 2 runs through a loop	Loop bound is $k \leq 2$

Data-Flow Coverage (Datenflussorientierter Test)

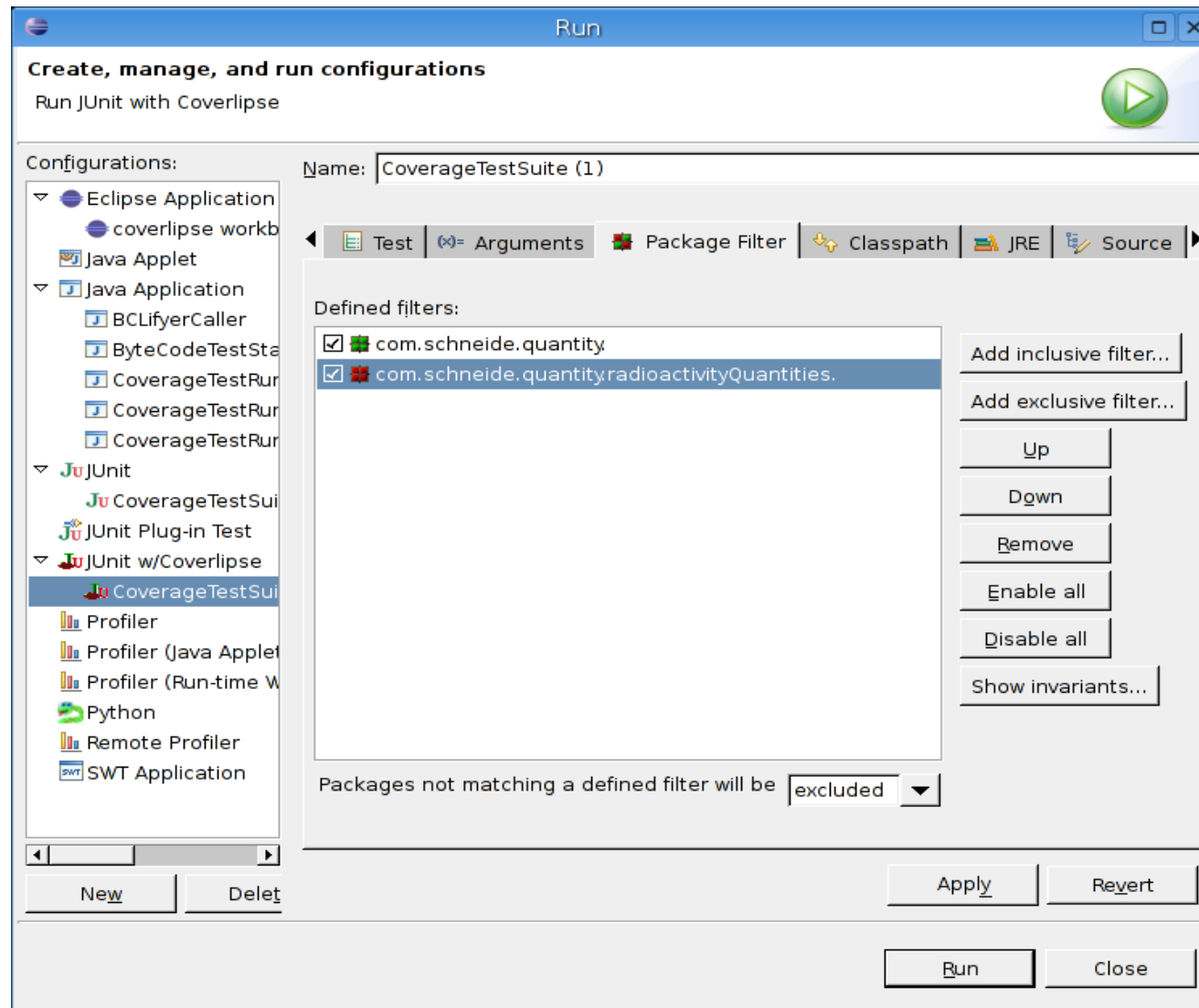
Coverage class	Technique	Purpose
All defs	For all definitions (assignments) of variables: one path to a use has to be tested	Discover dead variable assignments (definitions)
All p-uses	For a definition (assignment) of a variable, test all uses <i>in predicates</i>	Show the influence of the variable assignment to the control flow
All c-uses	For a definition (assignment) of a variable, test all uses <i>outside of predicates</i>	Show the influence of the variable assignment to the data flow

Ex.: Coverlipse based on JUnit

- ▶ Selection of Junit test cases and their path coverage analysis



Coverlipse- Selecting Packages to Analyze



Coverlipse Block Coverage / Statement Coverage

The screenshot displays the Eclipse IDE interface with the following components:

- Code Editor:** Shows the `BlockVarCombinator.java` file. The `equals` method is highlighted, with green checkmarks indicating full coverage for lines 25, 28, 31, and 32, and a red exclamation mark indicating no coverage for line 29.
- Coverlipse Class Coverage View:** Shows a tree of classes with coverage percentages. The selected class is `de.uka.ipd.coverage.ssa.helpers.BlockVarCombinator` with 89% coverage. Other classes include `MapBlockVarCombinatorToIS` (100%) and `MapVariableToInt` (100%).
- Problems View:** Shows a table of coverage messages.

Message	Li	cove	unco	Resource
✓ This line was fully covered	25			BlockVarCombinator.java
✓ This line was fully covered	28			BlockVarCombinator.java
! This line was not covered	29			BlockVarCombinator.java
✓ This line was fully covered	31			BlockVarCombinator.java
✓ This line was fully covered	32			BlockVarCombinator.java

Coverlipse: All-uses Data-Flow Coverage

The screenshot displays the Eclipse IDE interface with the Coverlipse plugin. The main editor shows the source code of the `Computation` class. The left sidebar shows the 'All-Uses Coverage' view with a tree structure containing 'simpletests' and 'Computation'. The bottom panel shows the 'Coverlipse Markers View' with a table of variable definitions.

```
public class Computation {  
    public int add(int arg1, int arg2) {  
        int result = arg1 + arg2; int meinInt = 0;  
        int result2 = result;  
        if (arg1 == Integer.MIN_VALUE) {  
            new Integer(result);  
        }  
        int result3 = result2;  
        return result + meinInt;  
    }  
  
    public int multiply(int n, int m) {  
        int result = 0;  
        for (int j = 0; j < m; j++) {
```

Message	Line	covered uses	uncovered uses	Resource
Definition of the variable arg1	12	13 15		Computation.java
Definition of the variable arg2	12	13		Computation.java
Definition of the variable meinInt	13	19		Computation.java
Definition of the variable result	13	14 19	16	Computation.java
Definition of the variable result2	14	18		Computation.java
Definition of the variable result3	18			Computation.java

Coverlipse: Problem Description of a Use of a Variable

The screenshot shows the Eclipse IDE with the Coverlipse plugin. The main editor displays the following Java code:

```
public class Computation {  
    public int add(int arg1, int arg2) {  
        int result = arg1 + arg2; int meinInt = 0;  
    }  
}
```

A context menu is open over the variable `result` on line 13. The menu items are:

- Show all definitions
- Show all definitions
- Show uses of the variable meinInt
- Show uses of the variable result

The selected item, "Show uses of the variable result", has a tooltip that reads: "Problem description: Definition of the variable result".

At the bottom of the IDE, a table displays the analysis results:

Message	Line	covered uses	uncovered uses	Resource
Definition of the variable arg1	12	13 15		Computation.java
Definition of the variable arg2	12	13		Computation.java
Definition of the variable meinInt	13	19		Computation.java
Definition of the variable result	13	14 19	16	Computation.java
Definition of the variable result2	14	18		Computation.java
Definition of the variable result3	18			Computation.java

Coverlipse: all-uses Coverage

```
public class Computation {  
    public int add(int arg1, int arg2) {  
        int result = arg1 + arg2; int meinInt = 0;  
        int result2 = result;  
        if (arg1 == Integer.MIN_VALUE) {  
            new Integer(result);  
        }  
        int result3 = result2;  
        return result + meinInt;  
    }  
  
    public int multiply(int n, int m) {  
        int result = 0;  
        for (int j = 0; j < m; j++) {
```

Message	Line	covered uses	uncovered uses	Resource
Definition of the variable meinInt	13	19		Computation.java
✓ Definition of the variable result	13	14 19	16	Computation.java
Definition of the variable result2	14	18		Computation.java
↓ This use was covered	14			Computation.java
✗ This use was not covered	16			Computation.java
Definition of the variable result3	18			Computation.java
↓ This use was covered	19			Computation.java



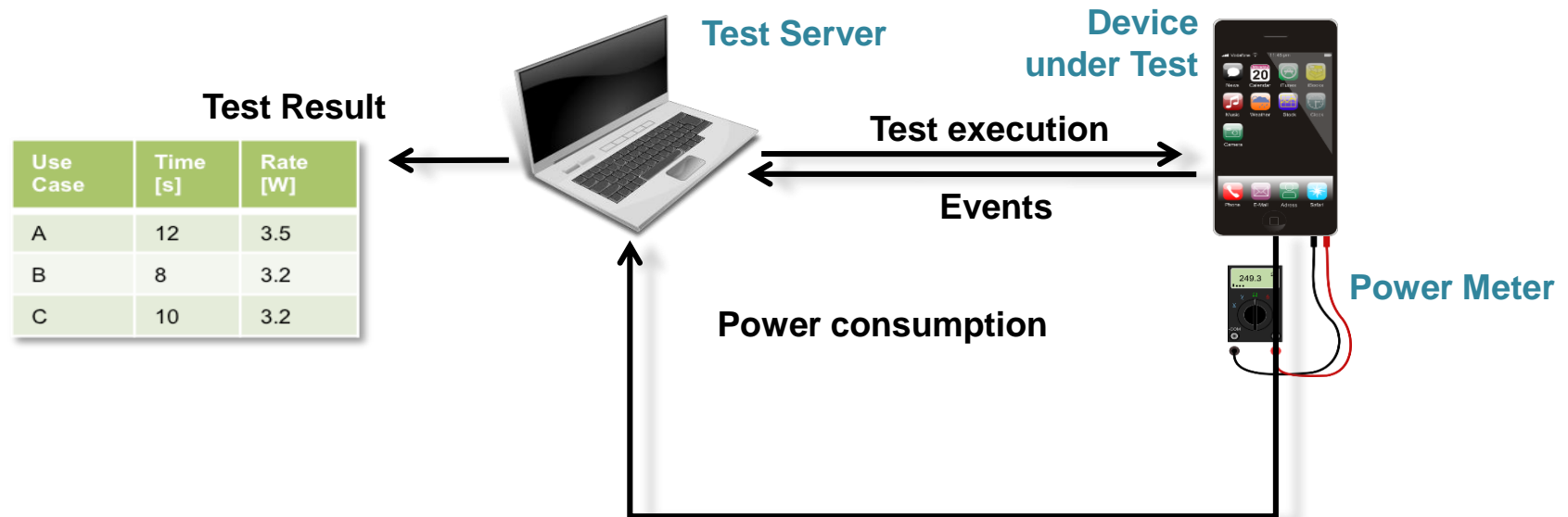
- JouleUnit (courtesy Claas Wilke)
- YouTube-Video: <http://is.gd/energyLabel>
<http://www.qualitune.org/>
<http://www.jouleunit.org/>



12.2.2 CODE-CENTRIC ENERGY TEST-FRAMEWORKS

Energy Test with JouleUnit

- ▶ Generic profiling framework [WGR13]
 - Based on unit tests with jUnit: Test cases define workloads
- ▶ Reusable for different platforms, e.g., Android, NAO robots



Energy Test

Softwaretechnologie II



JouleUnit + QMark

- ▶ Extension of JUnit for Energy tests of Android apps
- ▶ Reuse of Junit functional tests
- ▶ Execution on Smart Phone
 - Feedback on energy bugs
- Remote execution on Qmark test server
 - Hardware energy profiling



JouleUnit Workbench (Eclipse)

The screenshot displays the Eclipse IDE with the JouleUnit workbench. The Project Explorer on the left shows a project structure with various test cases. The Test Run Details window is active, showing a scatter plot of performance metrics over time. The plot includes data for Power Rate [mW], CPU Frequency [MHz], WiFi Traffic [Byte], and LCD Brightness [0..255]. The Test Run Results window shows a table of individual test case results, including start and stop times, duration, average power rate, and energy consumption. The Console window at the bottom shows the execution log for the test run.

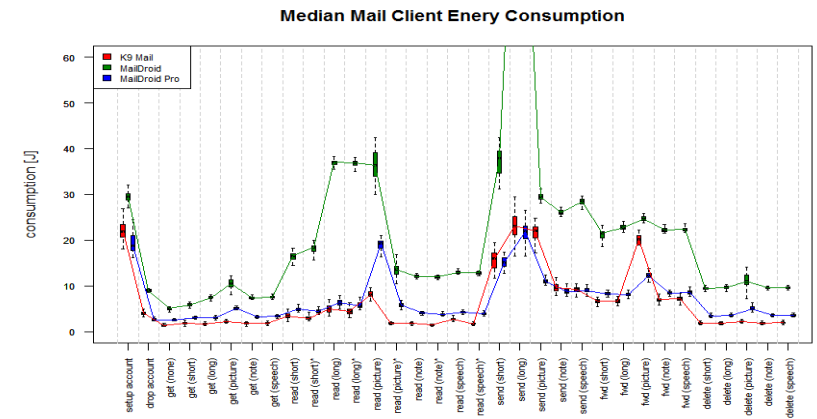
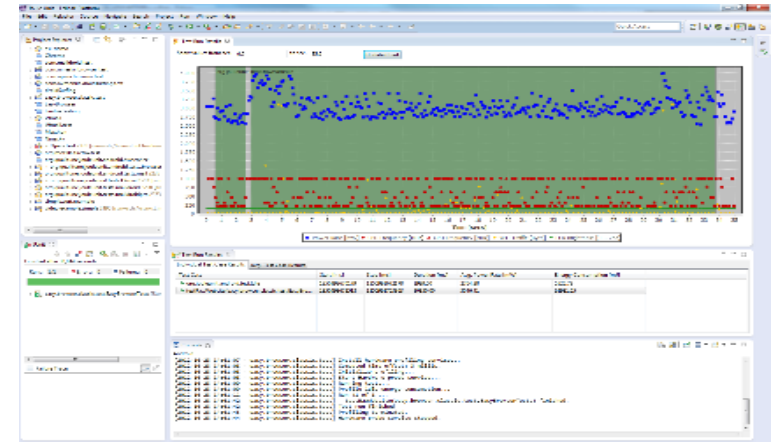
Test Case	Start [ms]	Stop [ms]	Duration [ms]	Avg. Power Rate [mW]	Energy Consumption [mJ]
org.jouleunit.android.test.Idle	1350993670179	1350993672178	1999,00	2764,28	5525,79
testRawWebsite(easy.browser.classic.test.EasyBro...	1350993672513	1350993703363	30850,00	3046,41	93981,65

```
Android
[2012-10-23 14:01:07 - easy.browser.classic.test] Install hardware profiling service...
[2012-10-23 14:01:08 - easy.browser.classic.test] Computed time offset: 3 millis.
[2012-10-23 14:01:08 - easy.browser.classic.test] Initialize profiling...
[2012-10-23 14:01:08 - easy.browser.classic.test] Start Hardware probe service...
[2012-10-23 14:01:09 - easy.browser.classic.test] Running tests...
[2012-10-23 14:01:09 - easy.browser.classic.test] Profile idle energy consumption...
[2012-10-23 14:01:11 - easy.browser.classic.test] Run #1 of 1 ...
[2012-10-23 14:01:42 - easy.browser.classic.test] testRawWebsite(easy.browser.classic.test.EasyBrowserTests) finished.
[2012-10-23 14:01:42 - easy.browser.classic.test] Test run finished
[2012-10-23 14:01:43 - easy.browser.classic.test] Profiling terminated.
[2012-10-23 14:01:44 - easy.browser.classic.test] Hardware probe service stopped.
```

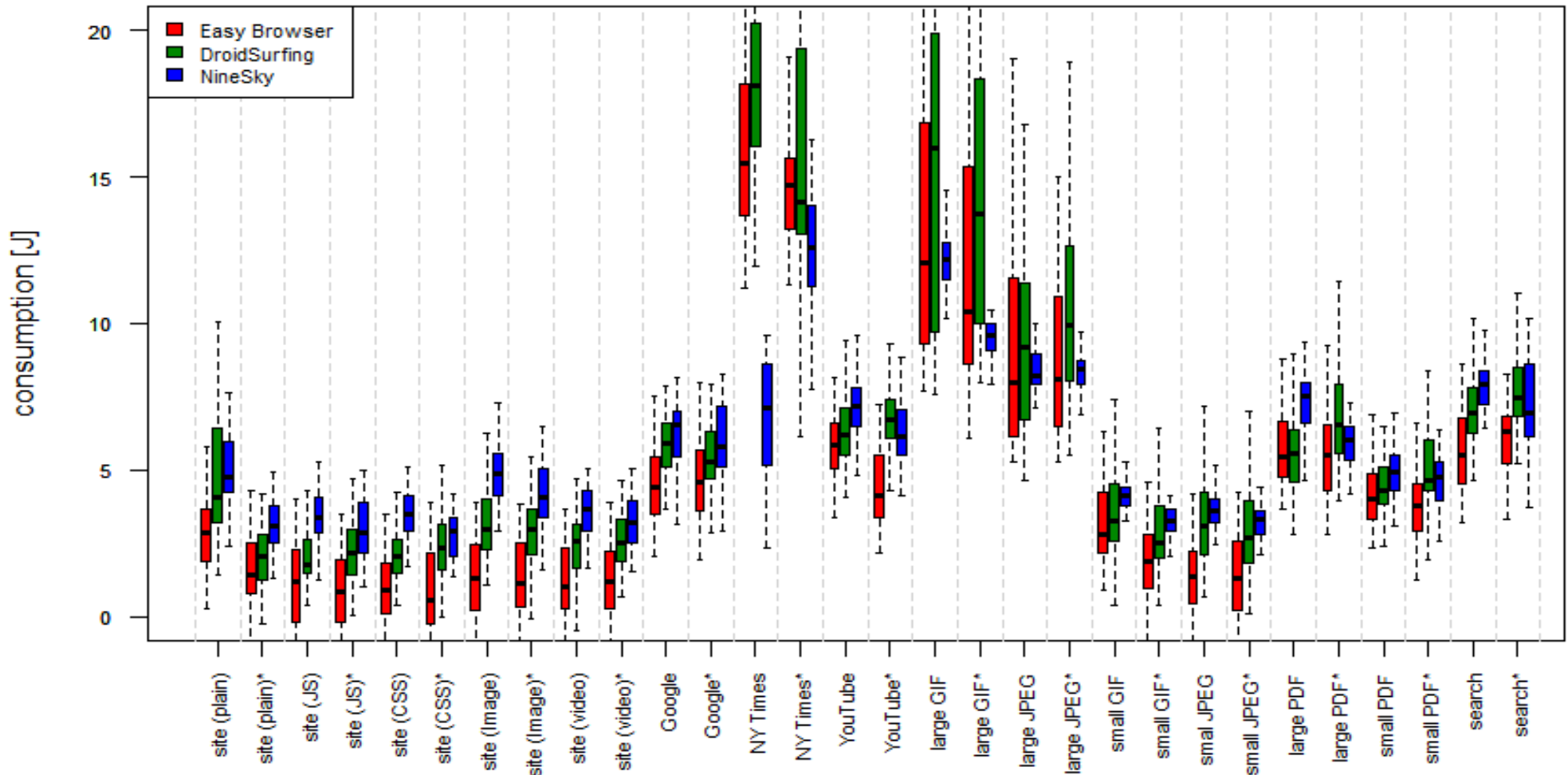


Compare „Simliar“ Apps

- ▶ Definition of benchmarks
 - Web browsing
 - Emailing
- ▶ Apps:
 - EasyBrowser, DroidSurfing, NineSky
 - K9 Mail, MailDroid, MailDroid Pro

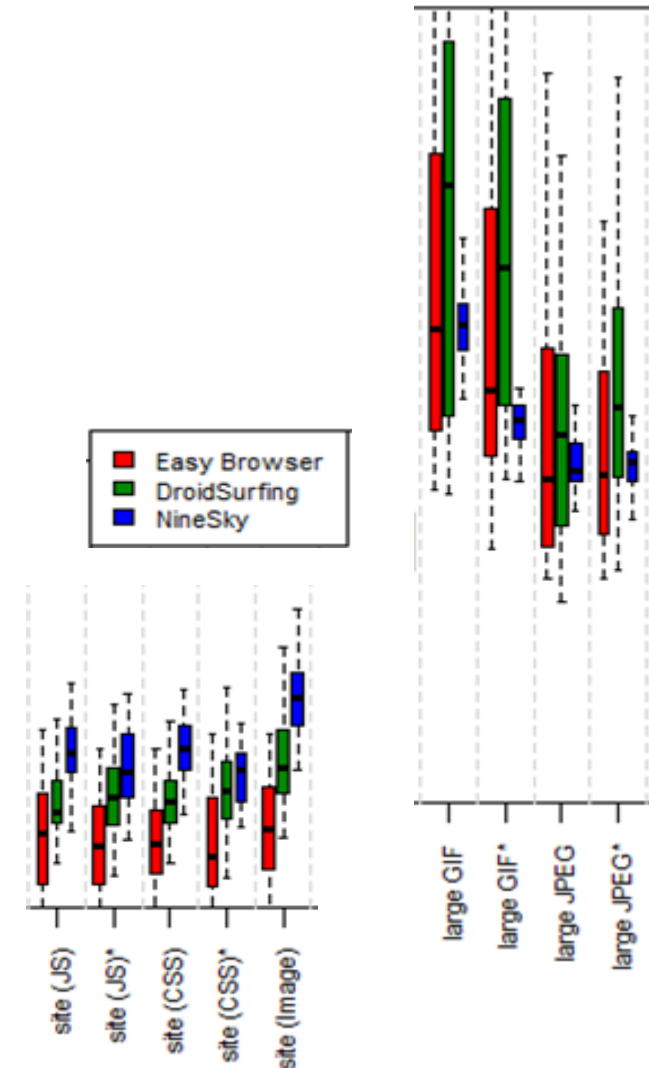


Median Web Browser Energy Consumption

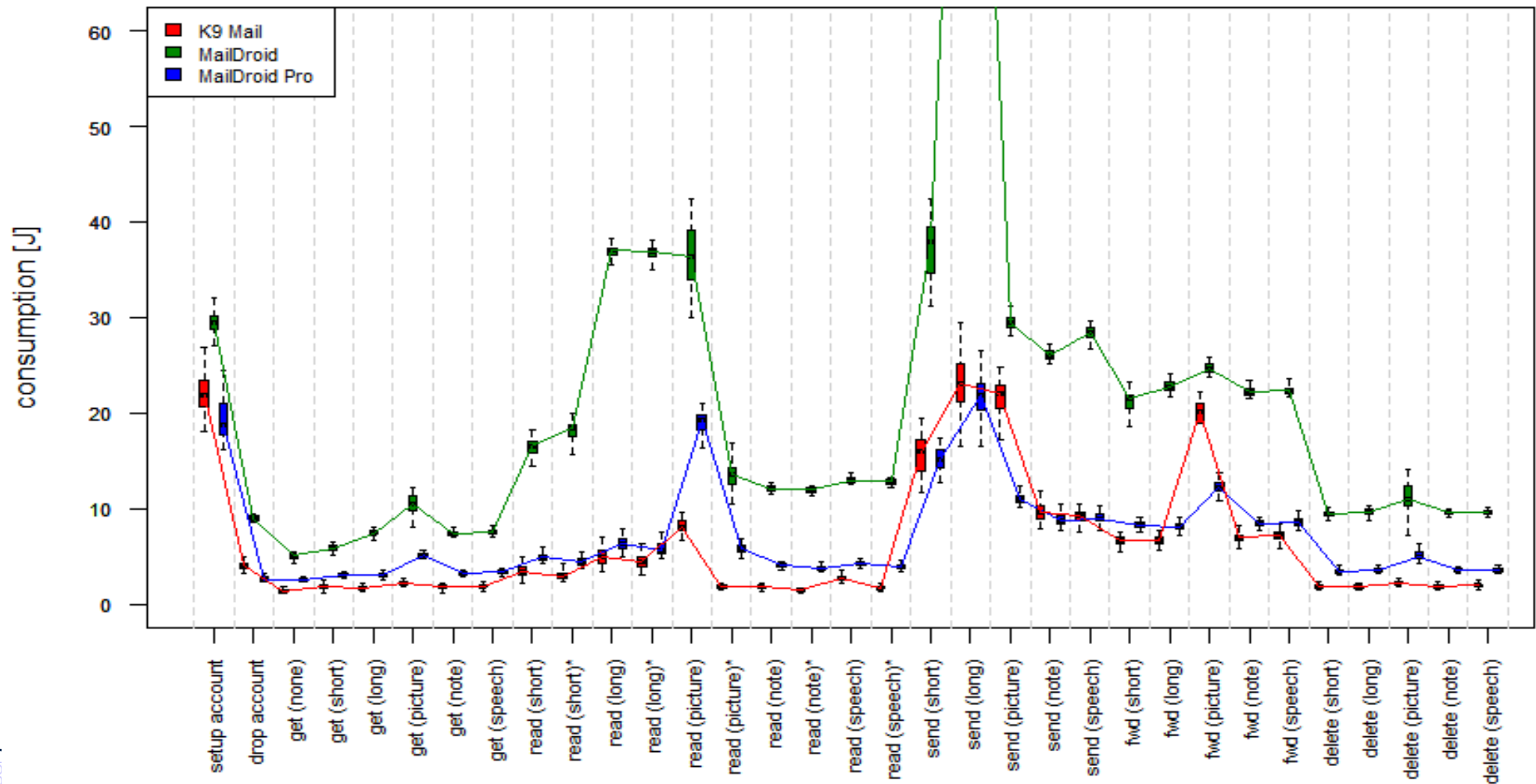


Interesting Issues with Web Browsers

- ▶ High variance in measurements (by high variance of response times)
 - But: comparable trends
- ▶ NineSky worst for small pages
 - ▶ But better for big images (because faster)
- ▶ Advertisement in EasyBrowser, DroidSurfing has negative influence only during long load times
- ▶ Different browsers are optimal for different use cases

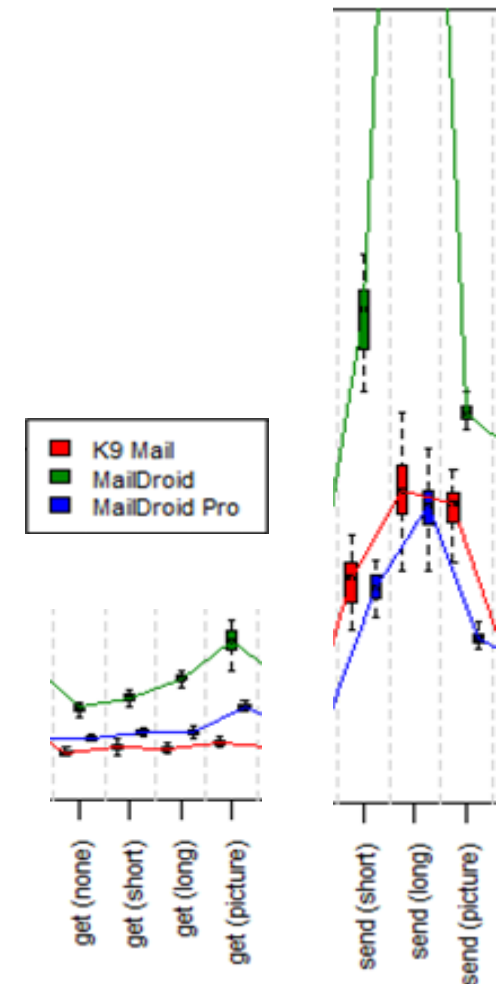


Median Mail Client Energy Consumption



Interesting Issues with Email Clients

- ▶ Low variance of measurements
- ▶ MailDroid worst for all scenarios
 - Reason: Advertisement
 - Negative influence grows for long scenarios
 - MailDroid pro & K9 Mail are similar
- ▶ Differences in energy consumption
- ▶ Avoid advertisement



12.2.3 ECLIPSE-BASED TEST PLATFORMS

Eclipse Supports Many Test Platform Plugins

- Hyades www.eclipse.org/test-and-performance
 - Test Capture-Replay framework for web and other applications
 - Http requests can be recorded, generated into JUnit test case classes, afterwards replayed
 - Uses UTP to specify test cases
 - A Remove-Access-Controller mediates between Eclipse and the SUT
 - Test data can be stored in data pools
 - Log-file analysis based on the Common-Base-Event format of IBM
- Solex http proxy logger www.sf.net/projects/solex
- Scapa stress test www.scapatech.com
- HttpUnit, htmlUnit extensions of JUnit for test of web applications
 - httpunit.sf.net
 - htmlunit.sf.net

- TPTP Platform Project <http://www.eclipse.org/tptp/>
 - Covers the common infrastructure in the areas of user interface, EMF based data models, data collection and communications control, remote execution environments and extension points
- TPTP Monitoring Tools Project
 - Collects, analyzes, aggregates and visualizes data that can be captured in the log and statistical models
- TPTP Testing Tools Project
 - Provides specializations of the platform for testing and extensible tools for specific testing environments
 - 3 test environments: JUnit, manual and URL testing
- TPTP Tracing and Profiling Tools Project
 - Extends the platform with specific data collection for Java and distributed applications that populate the common trace mode, also viewers and analysis services

TPTP Profiling Tool

Profiling and Logging - ProductCatalog.java - Eclipse SDK

File Edit Source Refactor Navigate Search Project Run Window Help

Working Sets

Console Execution Statistics Method Invocation Details

Execution Statistics - com.sample.product.Product at popescu011 [PID: 396] (Filter: No filter)

Method	Class	Package	Base Time (sec...)	<Cumulative Ti...	Calls
main(java.lang.String[]) void	Product	com.sample.p...	0.07%	44.68%	0.02%
readData(java.lang.String) void	ProductCatalog	com.sample.p...	0.05%	41.08%	0.02%
parseContent(java.io.File, javax.xml.parsers.SAXParser) void	ProductCatalog	com.sample.p...	0.09%	21.30%	0.39%
createParser() javax.xml.parsers.SAXParser	ProductCatalog	com.sample.p...	0.02%	19.34%	0.39%
parse(java.io.InputStream, org.xml.sax.InputSource) void	SAXParser	javax.xml.pa...	0.07%	19.23%	0.39%
parse(org.xml.sax.InputSource, org.xml.sax.InputSource) void	SAXParser	javax.xml.pa...	0.15%	19.11%	0.39%
parse(org.xml.sax.InputSource) void	AbstractSAXP...	org.apache.x...	13.02%	18.18%	0.39%
newInstance() javax.xml.parsers.SAXParserFactory	SAXParserFa...	javax.xml.pa...	0.04%	12.64%	0.02%
find(java.lang.String, java.lang.String) javax.xml.parsers.FactoryFinder	FactoryFinder	javax.xml.pa...	0.04%	12.53%	0.02%
findJarServiceProvider(java.lang.String) javax.xml.parsers.FactoryFinder	FactoryFinder	javax.xml.pa...	12.35%	12.35%	0.02%
newSAXParser() javax.xml.parsers.SAXParserFactory	SAXParserFa...	org.apache.x...	0.07%	6.64%	0.02%
SAXParserImpl(javax.xml.parsers.SAXParserFactory) javax.xml.parsers.SAXParserImpl	SAXParserImpl	org.apache.x...	0.32%	6.57%	0.02%
SAXParser() javax.xml.parsers.SAXParser	SAXParser	org.apache.x...	6.22%	6.22%	0.02%
startElement(java.lang.String, java.lang.String, java.lang.String) void	ProductCatalog	com.sample.p...	1.28%	5.17%	0.78%
println(java.lang.String) void	ConsolePrintS...	com.ibm.jvm.io	0.01%	3.18%	0.02%
println(java.lang.String) void	PrintStream	java.io	0.00%	3.00%	0.02%
newLine() void	PrintStream	java.io	2.89%	2.89%	0.02%
append(java.lang.String) java.lang.StringBuffer	StringBuffer	java.lang	1.40%	2.08%	12.23%
FileInputStream(java.io.File) java.io.FileInputStream	FileInputStream	java.io	0.13%	1.90%	0.39%
open(java.lang.String) void	FileInputStream	java.io	1.73%	1.73%	0.39%
StringBuffer(java.lang.String) java.lang.StringBuffer	StringBuffer	java.lang	0.41%	0.95%	2.27%

21M of 40M

Test Plugins for Eclipse

- ▶ <http://c2.com/cgi/wiki?CodeCoverageTools>

		Source	URL
Coverlipse	Simple coverage tool	Eclipse-based	coverlipse.sourceforge.net
EclEmma		Eclipse-based	http://eclemma.org/userdoc/coverageview.html



- A requirements-oriented test environment allows for
 - Tracing test cases to requirement specifications
 - Measuring the maturity of test with regard to the acceptance tests

12.3 REQUIREMENTS-ORIENTED TESTING

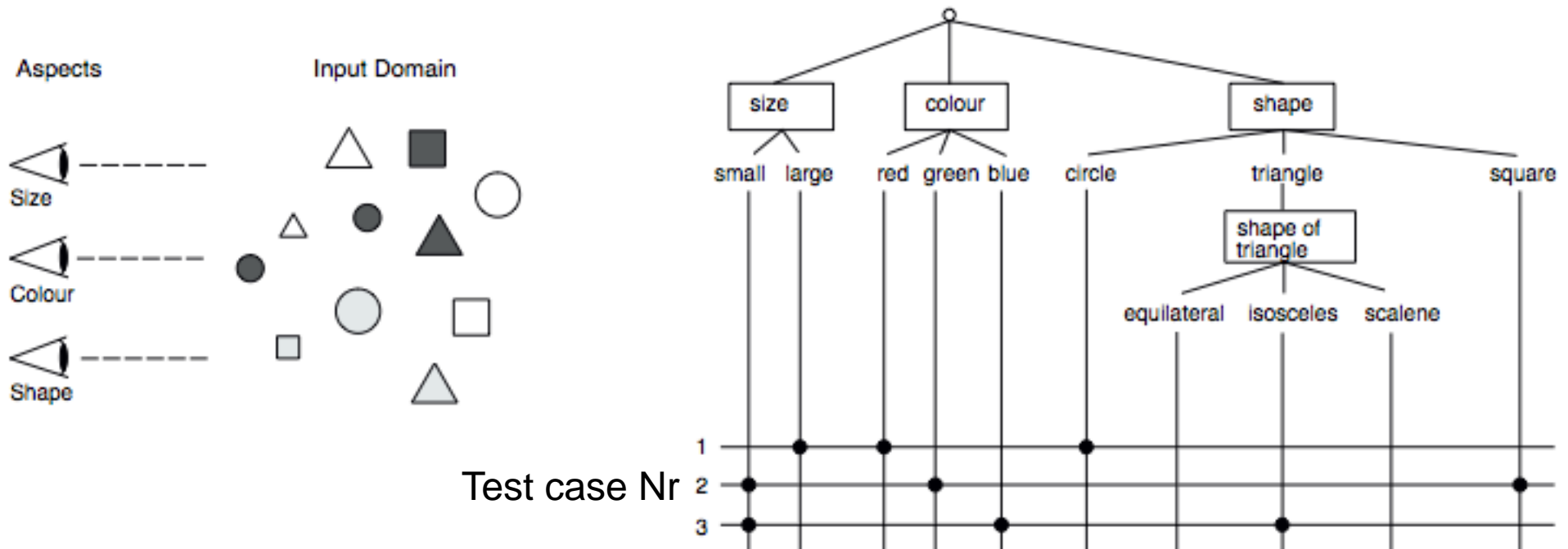
Test Environments

	Enterprise	URL
SilkTest	Segue Software	www.segue.com
TestBench	Imbus	www.imbus.de
Visual 2000	McCabe & Associates, USA	www.mccabe.com
Cantata++	IPL, Bath, UK	www.iplbath.com
ClickTracks	ClickTracks Analytics, Inc.CA	www.clicktracks.com
Tracetronic ECUTest	Tracetronic, Dresden. For motor tests	www.tracetronic.de

12.3.1 CLASSIFICATION TREE METHOD AND TESSY

Categories (Facets, Aspects) of the Test Case Data

- ▶ Test cases for testee objects and testee procedures are worked out in different categories (aspects, facets)
- ▶ Values of the parameters of testees are recursively divided into
 - ▶ intervals with one representant
 - ▶ Atomic value
- ▶ The values are grouped to test cases for a *test case table*

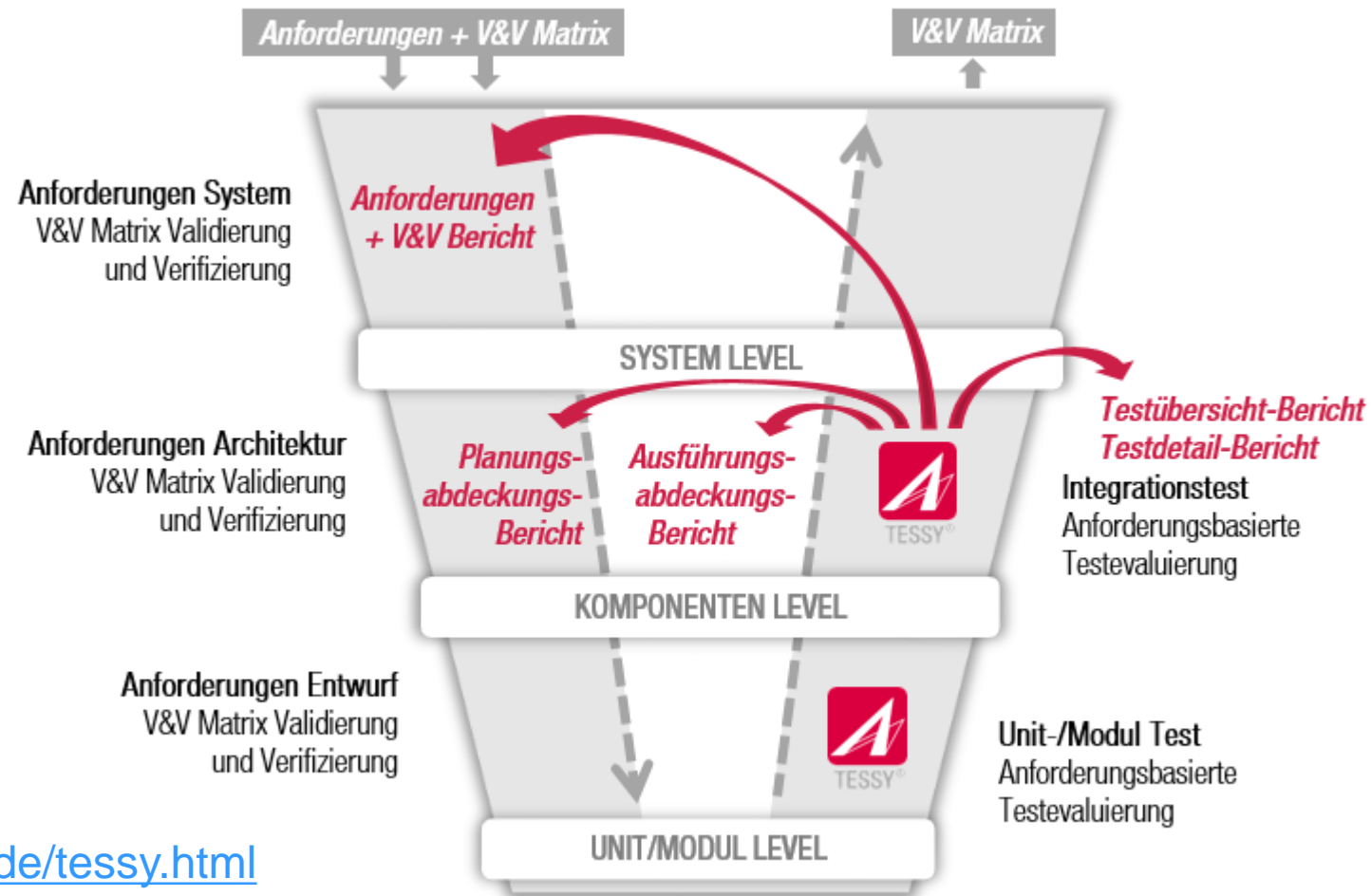


Advantages of Classification Tree Method

- ▶ http://www.hitex.com/fileadmin/pdf/products/tessy/white-papers/WP-TESSY-0102-CTE_e.pdf
- ▶ Division into categories reduces complexity of testing
 - Dimensional decomposition of parameter value space
 - Good visualization
- ▶ Representants of intervals should be chosen as boundary values of the intervals
- ▶ Coverage
 - ▶ A good combination of parameter values should cover the most important error cases
 - ▶ Generation of test cases

TESSY of HITEX

- ▶ Defining test cases with the classification tree method for regression tests
- ▶ Combination with coverage test



<http://www.hitex.com/>

<https://www.razorcat.com/de/tessy.html>

TESSY of HITEX

The screenshot displays the TESSY software interface for a project named 'Example-Project'. The main window is the 'Classification Tree Editor (CTE)', which shows a hierarchical tree structure for the component 'MEAS_CalcAverage'. The tree is divided into two main branches: 'Pointer to recorded values (cRecords)' and 'Count of recorded values (uclength)'. The 'Pointer to recorded values' branch is further divided into 'Invalid' and 'Valid'. The 'Valid' branch leads to a 'Recorded values (cRecords[])' node, which is further divided into 'Min.' and 'Max.' nodes. The 'Min.' node is further divided into '-128..127' and '0' nodes. The 'Max.' node is further divided into '-5..40' and 'Max.' nodes. The 'Count of recorded values' branch is further divided into 'Min.' and 'Max.' nodes. The 'Min.' node is further divided into '3' and '5' nodes. The 'Max.' node is further divided into 'Max.' and 'Max.' nodes. The right pane shows the 'Test Data of MEAS_CalcAverage' window, which displays a table of test data. The table has columns for 'Inputs', 'Parameters', 'Dynamics', and 'Outputs'. The 'Inputs' column shows 'char * cRecords' and 'unsigned char uclength'. The 'Parameters' column shows 'char target_cRecords[3]'. The 'Dynamics' column shows 'char target_cRecords[0]', 'char target_cRecords[1]', 'char target_cRecords[2]', 'char target_cRecords[3]', and 'char target_cRecords[4]'. The 'Outputs' column shows 'unsigned char ErrorHandler(error)'. The bottom status bar shows 'Project Root: T:\Projek\Exampl'.



12.3.2 IMBUS TESTBENCH

imbus TestBench

- Imbus TestBench is a test environment supporting
 - Test planning
 - Test analysis (connection of test cases to requirements)
 - Test design
 - Test automation (realization)
 - Test metrics and reports

<http://www.imbus.de/produkte/imbus-testbench/hauptfunktionen/>

Test-Status eines Requirements (rot, gelb, grün)

The screenshot shows a software interface for requirements management. The title bar reads "Anforderungsverwaltung von Car Konfigurator (Version 2.1, Abnahmetest)".

Anforderungsbaum:

- CarConfigurator - Version 1.1 (caliber)
 - 1. Business Requirements
 - Konfiguration zusammenstellen (Yellow)
 - Rabatt gewähren (Green)
 - automatische Rabatte (Green)
 - Händler gewährt Rabatt (Green)
 - 2. User Requirements
 - ständige Preisanzeige (Green)
 - keine erzwungene Bedienerfolge (Yellow)
 - 3. Functional Requirements
 - sofortige Preisberechnung (Red)
 - Quelle der Basisdaten (Green)
 - Import einer Datei (Green)
 - Import vom OEM-Host (Green)
 - 4. Design Requirements
 - gültige Konfiguration (Grey)
 - Eingabe der Basisdaten (Red)

Details Panel:

- Name:** Händler gewährt Rabatt
- ID:** WHY162
- Version:** 1.1
- Eigentümer:**
- Status:** Review Complete
- Priorität:** Essential
- Test-Status:** ■ Getestet PASS

Testf[...]: endpreis-berechnen-mit-rabatten_log.xml
Aktuelle Ansicht : Endpreis berechnen mit Rabatten : [...]jurieren : Fahrzeug wählen CBR
Menü ? - x

2.3.2 Endpreis berechnen mit Rabatten

- 1. einfach
 - CarConfig Starten
 - Preis prüfen
 - CarConfig Beenden
- 2. Testfall
 - CarConfig Starten
 - Fahrzeug konfigurieren
 - Fahrzeug wählen CBR**
 - Sondermodell wählen
 - Zubehör wählen
 - Preis prüfen
 - Fahrzeug konfigurieren
 - Fahrzeug wählen CBR
 - Sondermodell wählen
 - Zubehör wählen
 - Preis prüfen
 - Fahrzeug konfigurieren
 - Fahrzeug wählen CBR
 - Sondermodell wählen
 - Zubehör wählen
 - Preis prüfen
 - Endpreis berechnen "ohne" Rabatt
 - CarConfig Starten
 - Fahrzeug konfigurieren
 - Fahrzeug wählen CBR
 - Sondermodell wählen

Interaktion

Fahrzeug wählen CBR

Parameter	Wert
Fahrzeug	15

Fehler Fehler hinzufügen

Interaktion: Fahrzeug wählen CBR

-Beschreibung

Fahrzeug aus der Liste der Fahrzeuge wählen

Bemerkungen

-Bemerkungen zur Durchführung

-Bemerkungen zur Spezifikation

Benutzerdefinierte Felder der Durchführung

<für diesen Knotentyp können Benutzerdefinierte Felder nicht definiert werden>

Aufgezeichnete Attribute

Tester

Aktueller Benutzer

Tester

Letzte Änderung des Ergebnisses

Aktuelles Ergebnis Zu prüfen

Ergebnis-Datum (DD.MM.YYYY)

Ergebnis-Zeit (HH:MM:SS)

Zeitmessung

Geplante Durchführungszeit (DD:HH:MM:SS.SSS)

Aktuelle Durchführungszeit (DD:HH:MM:SS.SSS)

Liste der Anforderungen

Name	ID	Version	Eigentümer	Status	Priorität
sofortige Preisberechnung	WHAT303	3.1	Dierk	Accepted	Essential
keine erzwungene Bedienerfolge	USER302	1.0	Dierk	Submitted	Essential
ständige Preisanzeige	USER301	1.0	Dierk	Submitted	Essential



- Georg.Pueschel@tu-dresden.de

12.4 MODEL-DRIVEN TEST ENVIRONMENT MATE

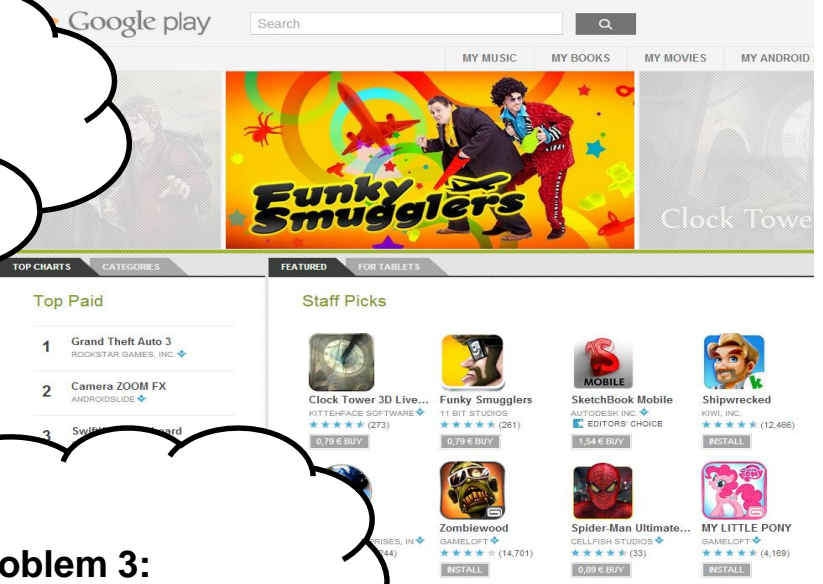
- **Model-driven testing** (MDT, MBT) generates black-box test cases from models, e.g., statecharts, petrinets, activity diagrams, sequence diagrams



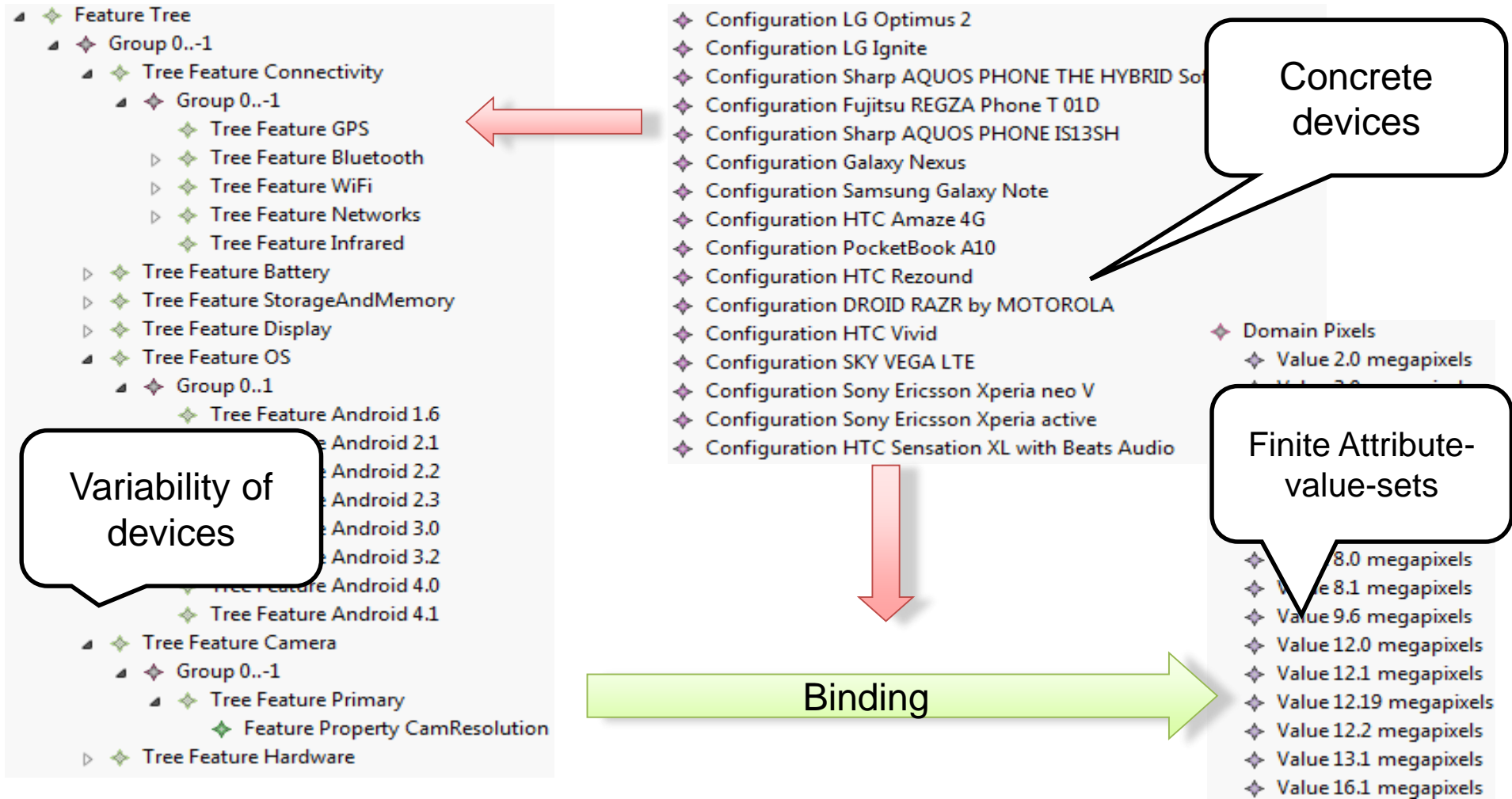
Problem 1:
Different platforms

Problem 2:
Apps differ on different devices

Problem 3:
Apps are context-adaptive



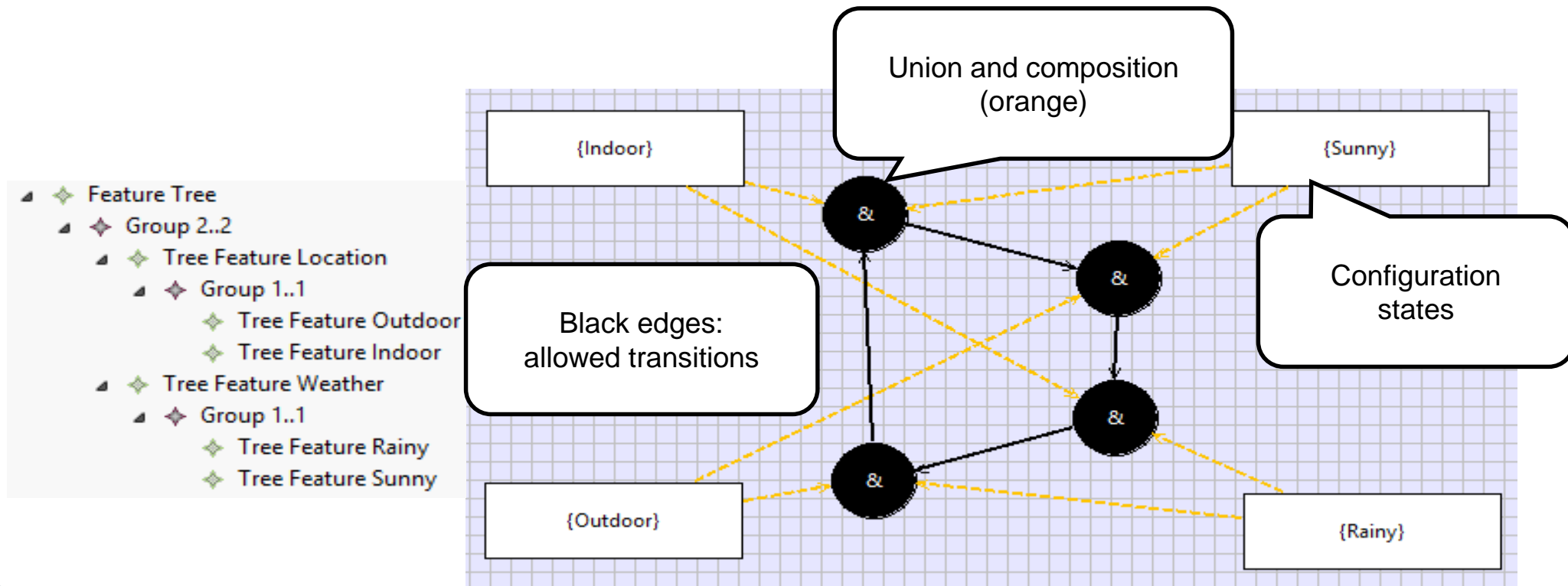
Plattform Management with Attributed Feature-Models (And-Or-Trees)



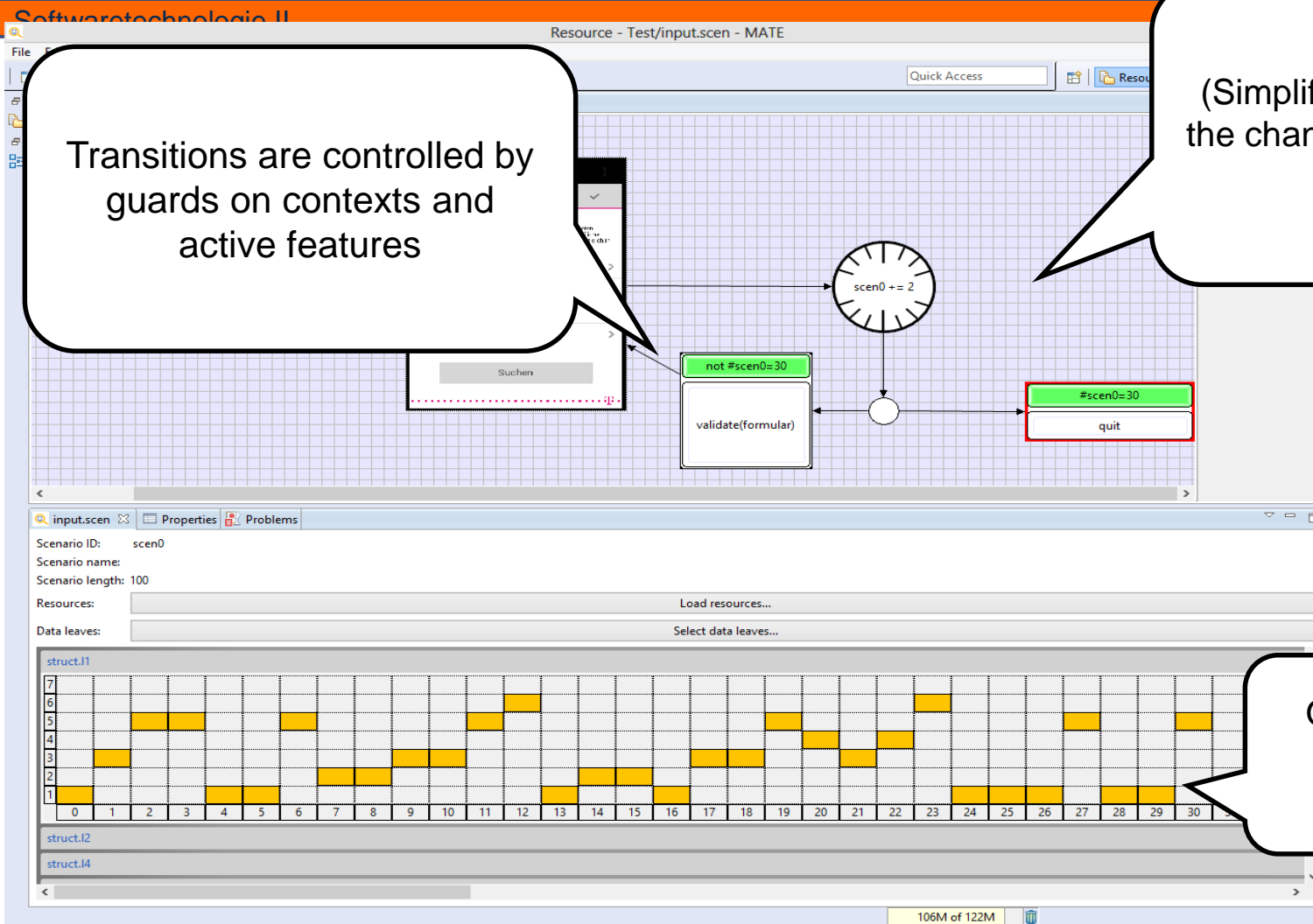
Variability in applications can be described by feature models

Dynamic Change of Features

- ▶ Features can be activated or deactivated at run time (**dynamic reconfiguration**)
 - ▶ Dynamic change of contexts based on a feature transition Petrinet
- ▶ Legal reconfigurations are described in a **operational configuration model (OCM)**



Specification of Legal Adaptations

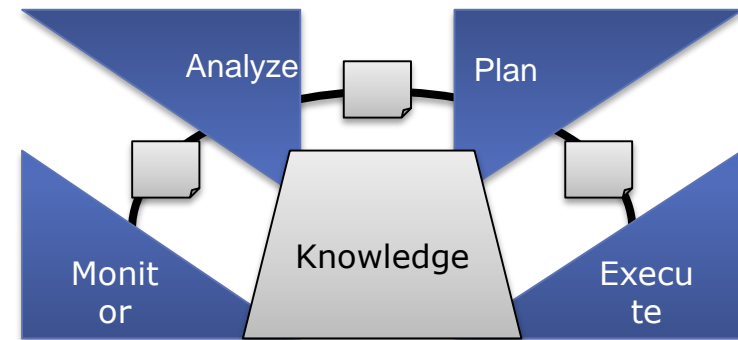


The MATE generator produces test cases by reachability analysis on the timed petri net.

Test of SAS must be Model-Based

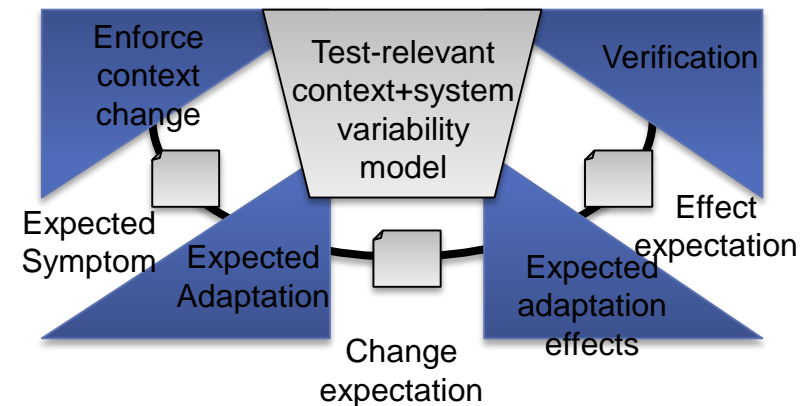
➤ Self-adaptive Systems (SAS) reconfigure themselves

- at runtime
- according to requirements
- And dependent on the „context“, which may change over time.



➤ In order to test SAS, one must

- stress the system with different contexts,
- alter contexts over time,
- model the system's expected adaptation
- and the expected effects of the adaptation on the system's behavior.



„counter feedback loop“ (CFL)

Automation: Coping with Complexity

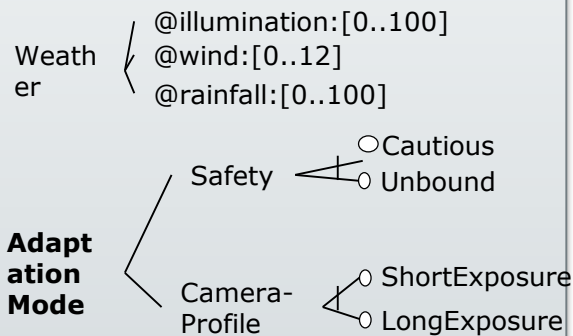
- **Solution A: Model-driven Testing (MDT)**
 - Black-box testing with automatic test design (in contrast to automatic test execution)
 - Test data, cases, and expected outcomes are generated from models
 - The models' expressiveness hides complexity
 - Adequacy criteria control generation

- **Solution B: SAS in the loop (ITL Simulation)**
 - MDT models are executed while the simulation state is compared to the real system one's for verification
 - ITL is more explorative as only one path through state space is considered during a single simulation
 - Enables testing reactions on non-controllable events (physical events that cannot be enforced, e.g. imprecise navigation of robots)

Model-based Adaptivity Test Environment (MATE)

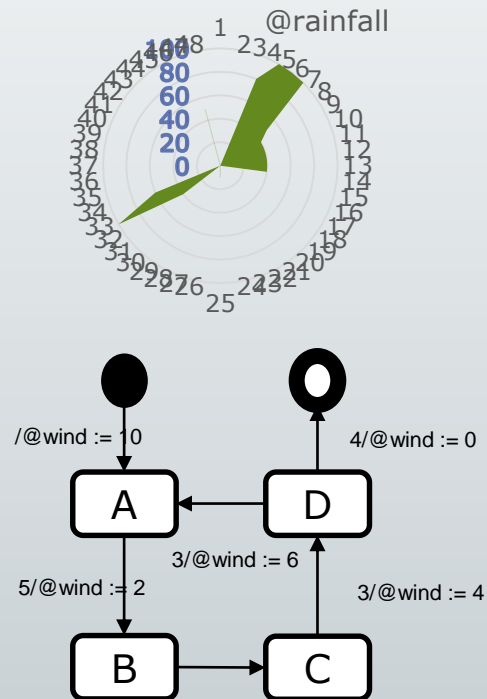
- implements both MDT and ITL for SAS
- provides SAS-specific **test metamodels**, interpreters, tooling, and a test adaptor framework

Context & System variability

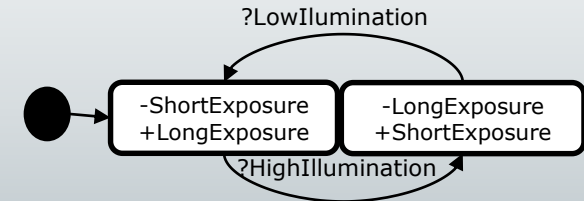


@wind > 4 and @rainfall > 20 ⇒ Cautious
 @illumination < 60 ⇒ LongExposure

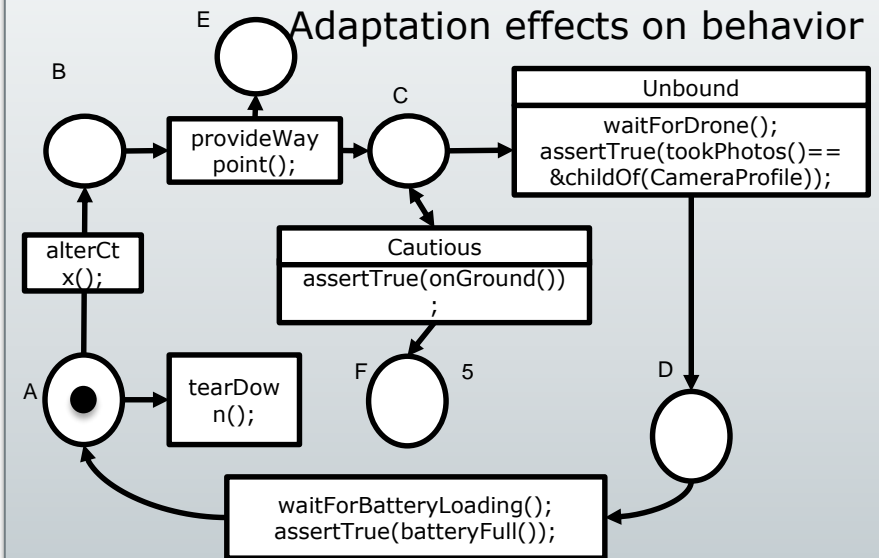
Context dynamics



Adaptation dynamics



Adaptation effects on behavior



MATE testing a self-adaptive production system transport system with robots

51 Softwaretechnologie II

The image displays two windows from the MATE (Model-based Adaptivity Test Environment) software. The left window, titled "Plant Overview - Operating mode - 'Demo-01'", shows a complex network of nodes and edges representing a production system. Nodes include "Goods in north 01", "Storage 01", "Storage 02", "Working station 01", "Working station 02", "Working station 03", and "Goods out 01". Edges represent transitions between these states, with labels like "Point-0027", "Point-0030", etc. The right window, titled "Resource - Test.gg - Model-based Adaptivity Test Environment", shows a test case diagram. The diagram consists of several nodes: a start node "gui503937...", three intermediate nodes (two labeled "produce('P1');" and "produce('P2');" with associated GUI IDs, and one labeled "sleep(1000); produce('P3');"), and an end node "gui813787...". The diagram uses various symbols for transitions, such as yellow rectangles for "true" conditions and blue ovals for nodes. Below the diagram, the "Progress" and "Simulation View" panels are visible. The "Simulation View" shows a list of simulation events, including "sleep(1000); + [instr 2 @sleep(100...)", "real_time(gui6461752148148322225) = 370442, token(gui2455374242249224099) = 2...", and "p(100...". The "Progress" panel shows the selected automation driver as "http://localhost" and the traversal strategy as "Random Traversal".



The End

Softwaretechnologie II

