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

C:\Program ...mindReviewer.ttp - IBM Rational Logiscope Viewer - machine_update_scores - Source - machine.c

File Edit Select Navigate View Options Window Help

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_upda
- make_code
- master/main
- play
- player_plays
- player_score
- print_help
- print_instructio
- print_instructio
- print_score_pl
- prompt
- refresh
- rest
- score_mac
- score_player
- set_dummy

machine_update_scores - Source - machine.c

```
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
```

machine_update_scores - Relative call graph

machine_update_scores - Control Graph

machine_update_scores - Criteria Kiviat Graph

CRITERION	CLASS
TESTABILITY	FAIR
STABILITY	GOOD
CHANGEABILITY	GOOD
ANALYZABILITY	FAIR
SYNTHESIS	FAIR

Works...

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

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

Press F1 for Help

RECEPTION

Logiscope Tools

- 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

© Prof. U. Aßmann

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)

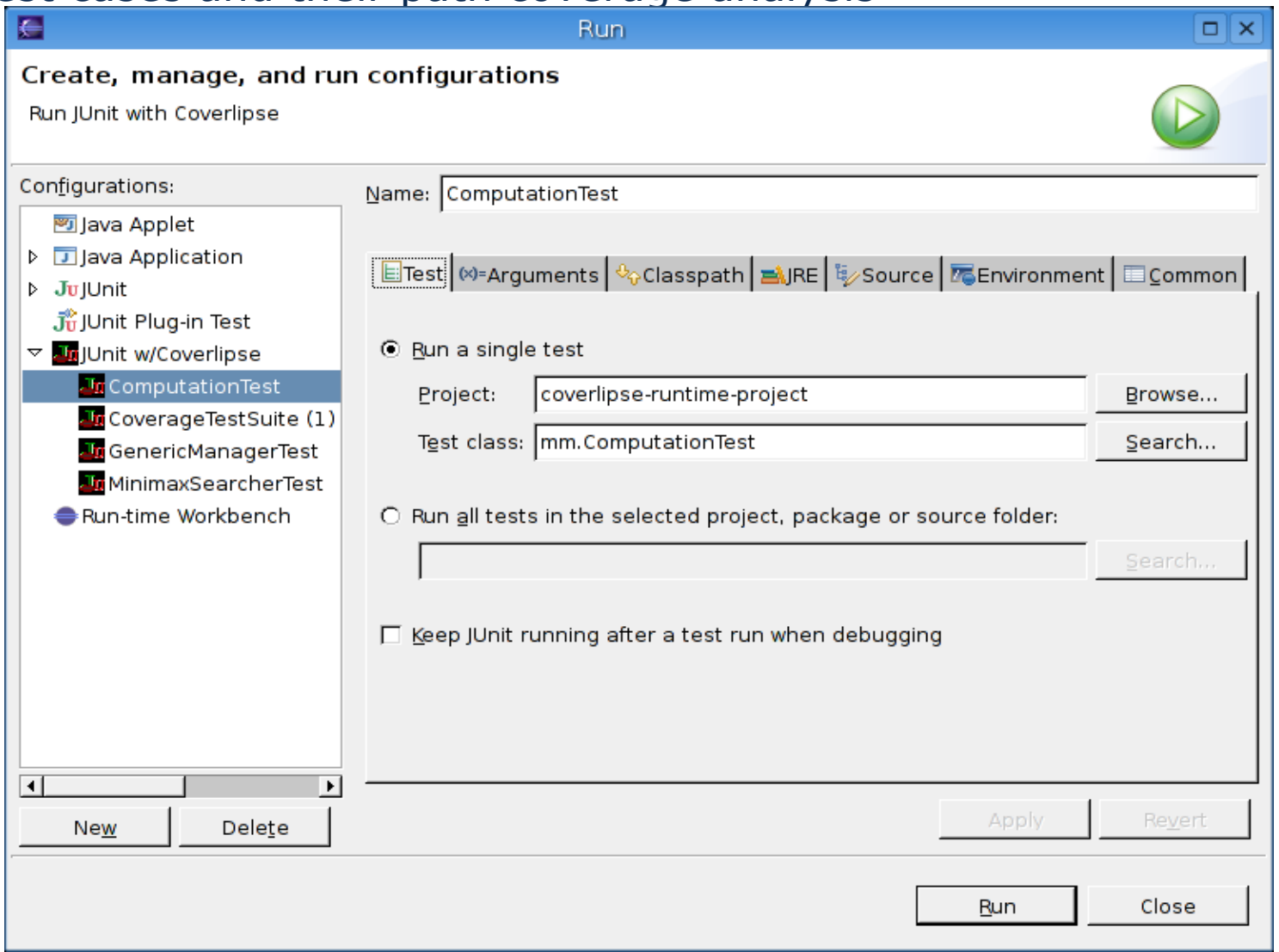
Coverage class	Technique	Purpose
Statement	How many statements are covered by how many test cases?	Discover dead code
Condition (Alternative)	How many branches are covered by how many test cases?	Cover all edges in control-flow graph
Combination of conditions	How many combinations of subsequent branches 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<=1
Interior Test	Coverage of all paths, with at most 2 runs through a loop	Loop bound is k<=2

Data-Flow Coverage (Datenflussorientierter Test)

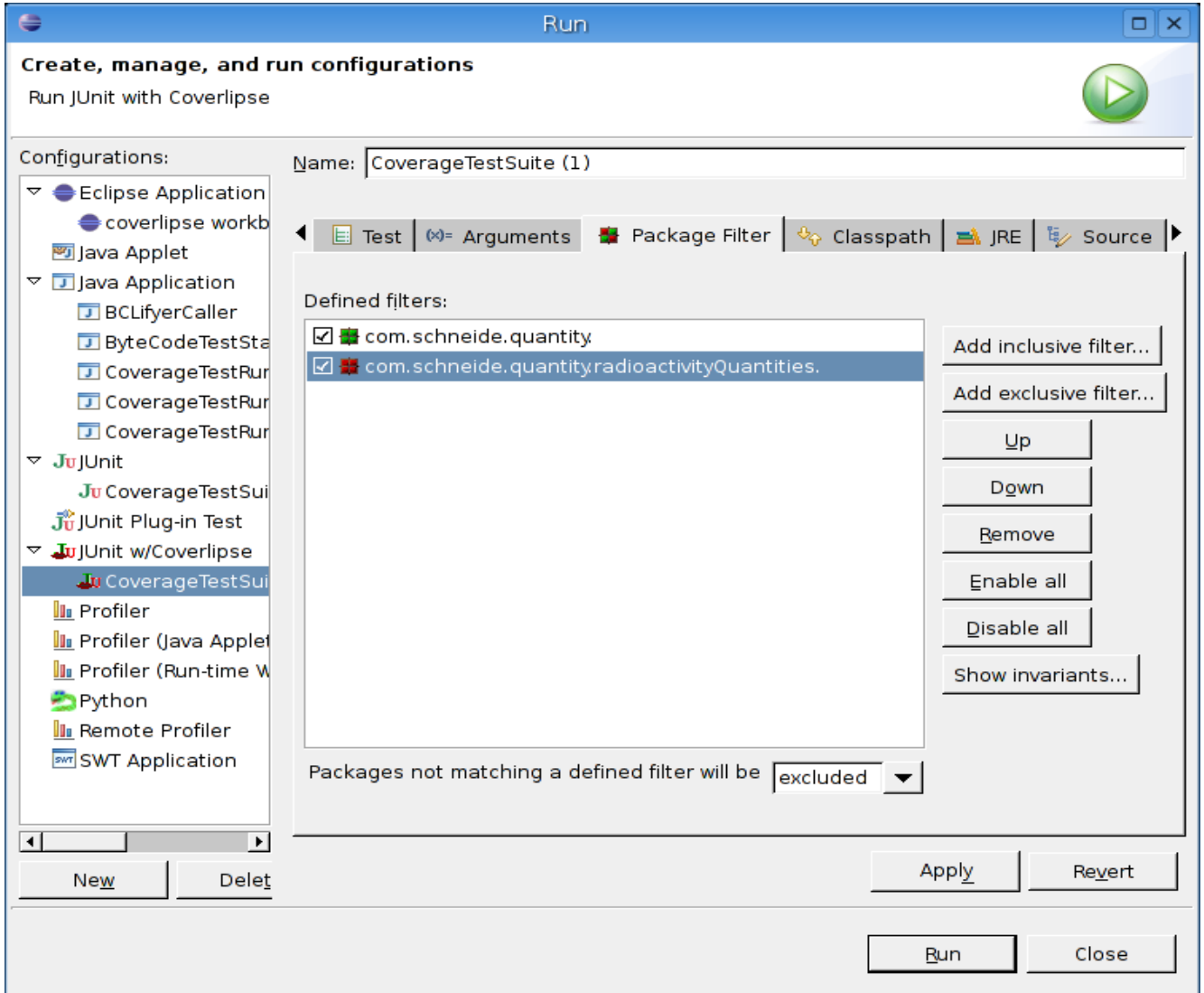
Coverage class	Technique	Purpose
All defs	For all definitions (assignments) of variables: one path to a use has to be tested	Discover dead variable assignments (definitions)
All p-uses	For a definition (assignment) of a variable, test all uses *in predicates*	Show the influence of the variable assignment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

Ex.: Coverlipse based on JUnit

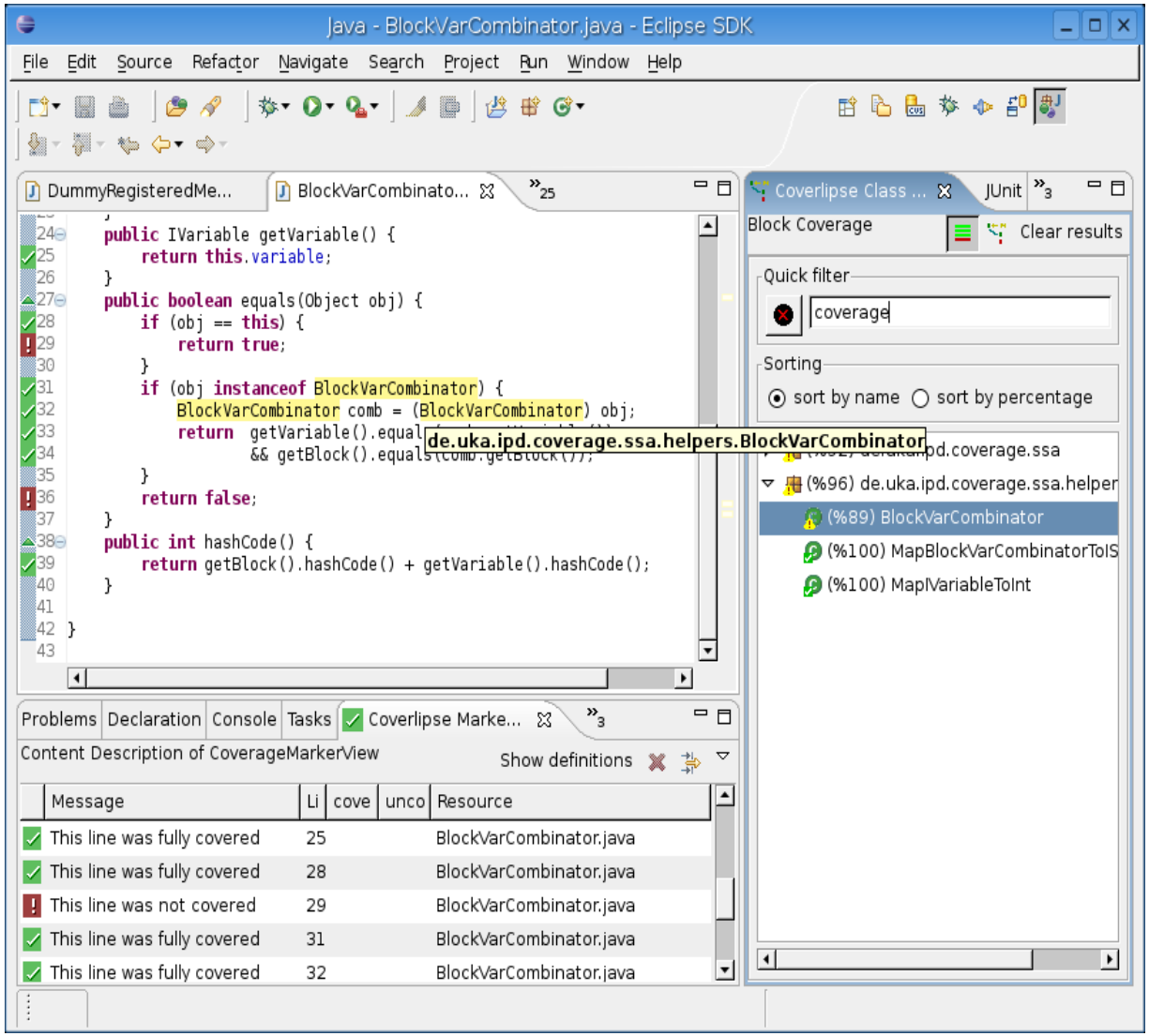
- Selection of Junit test cases and their path coverage analysis



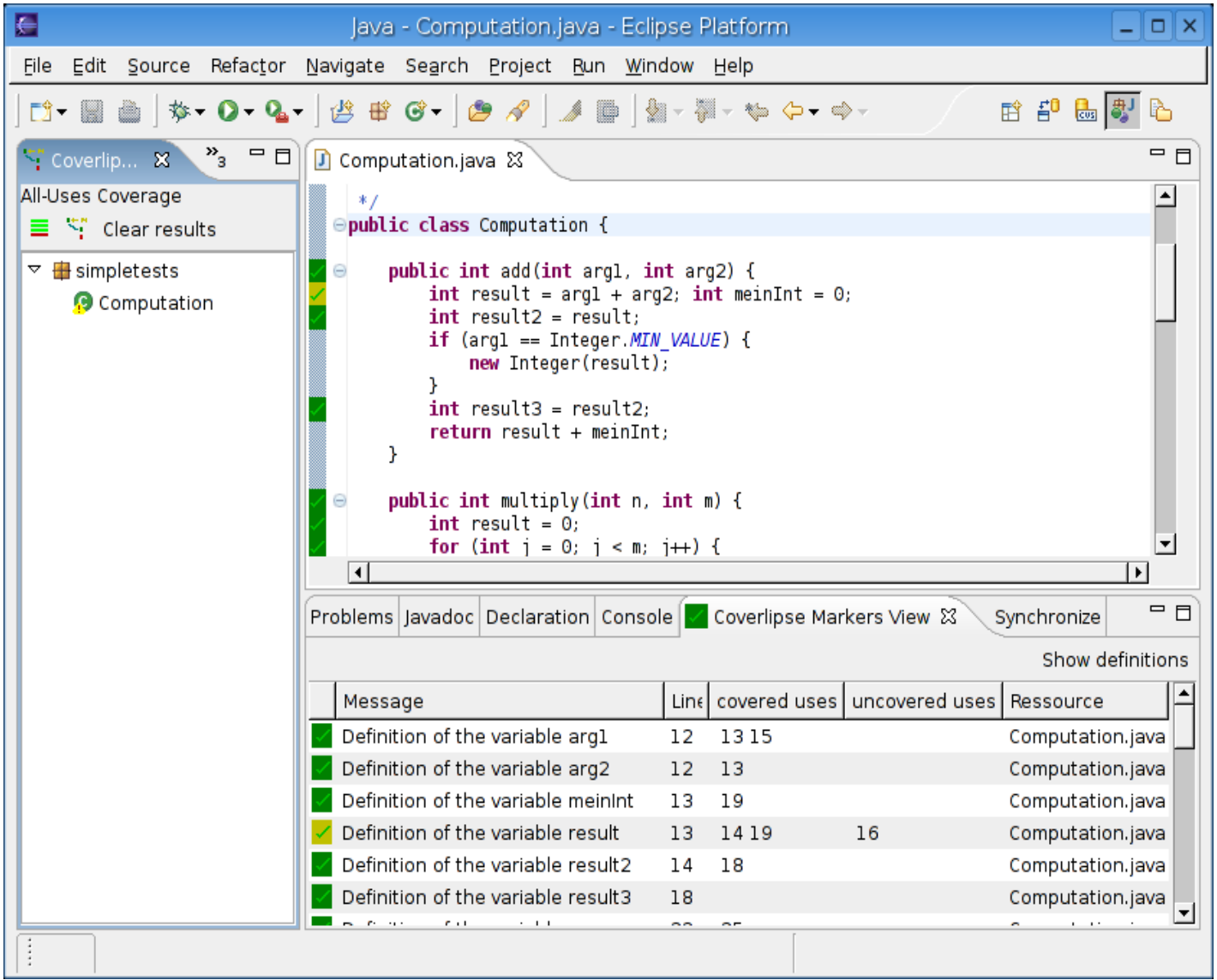
Coverlipse– Selecting Packages to Analyze



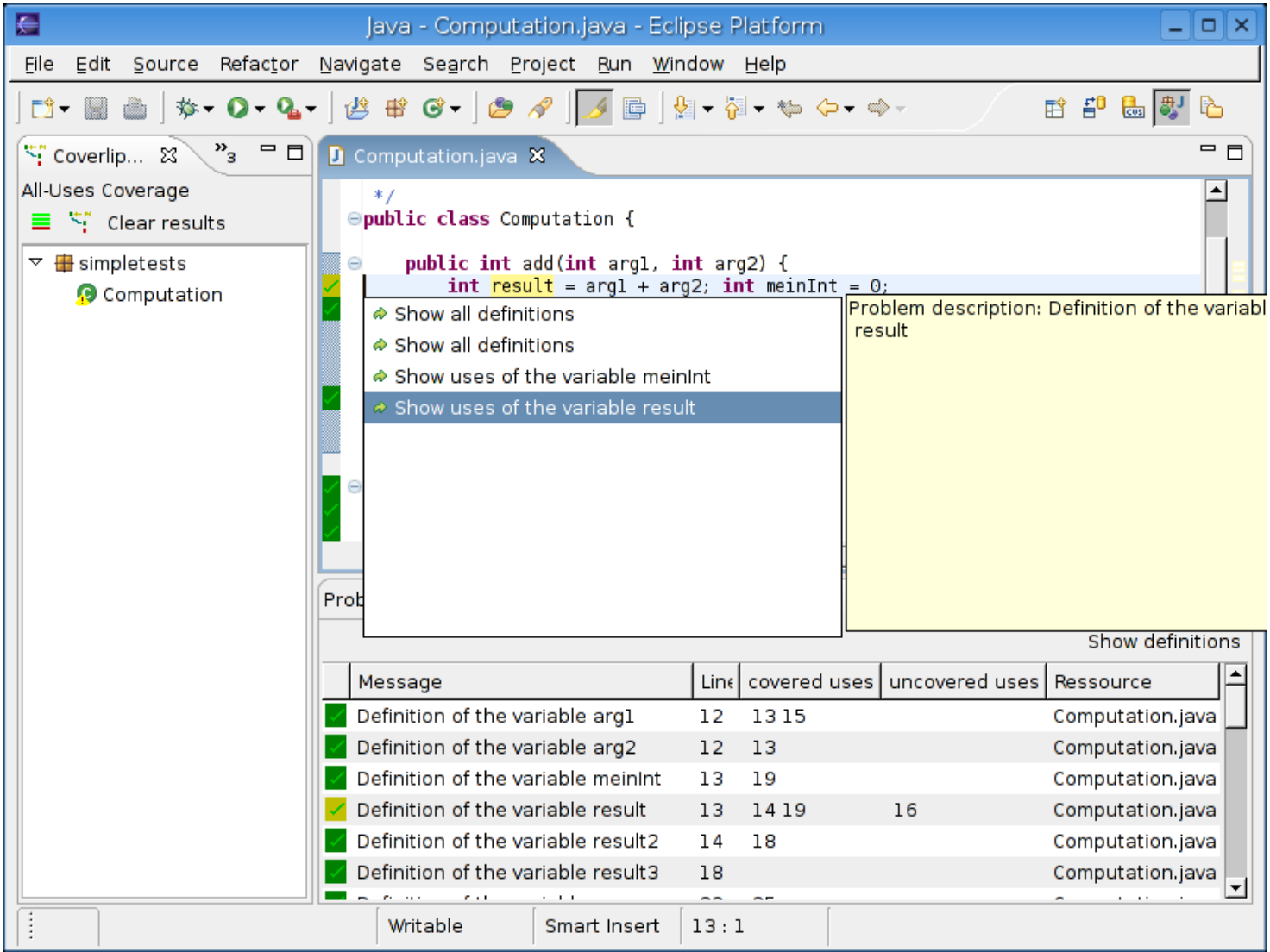
Coverlipse Block Coverage / Statement Coverage



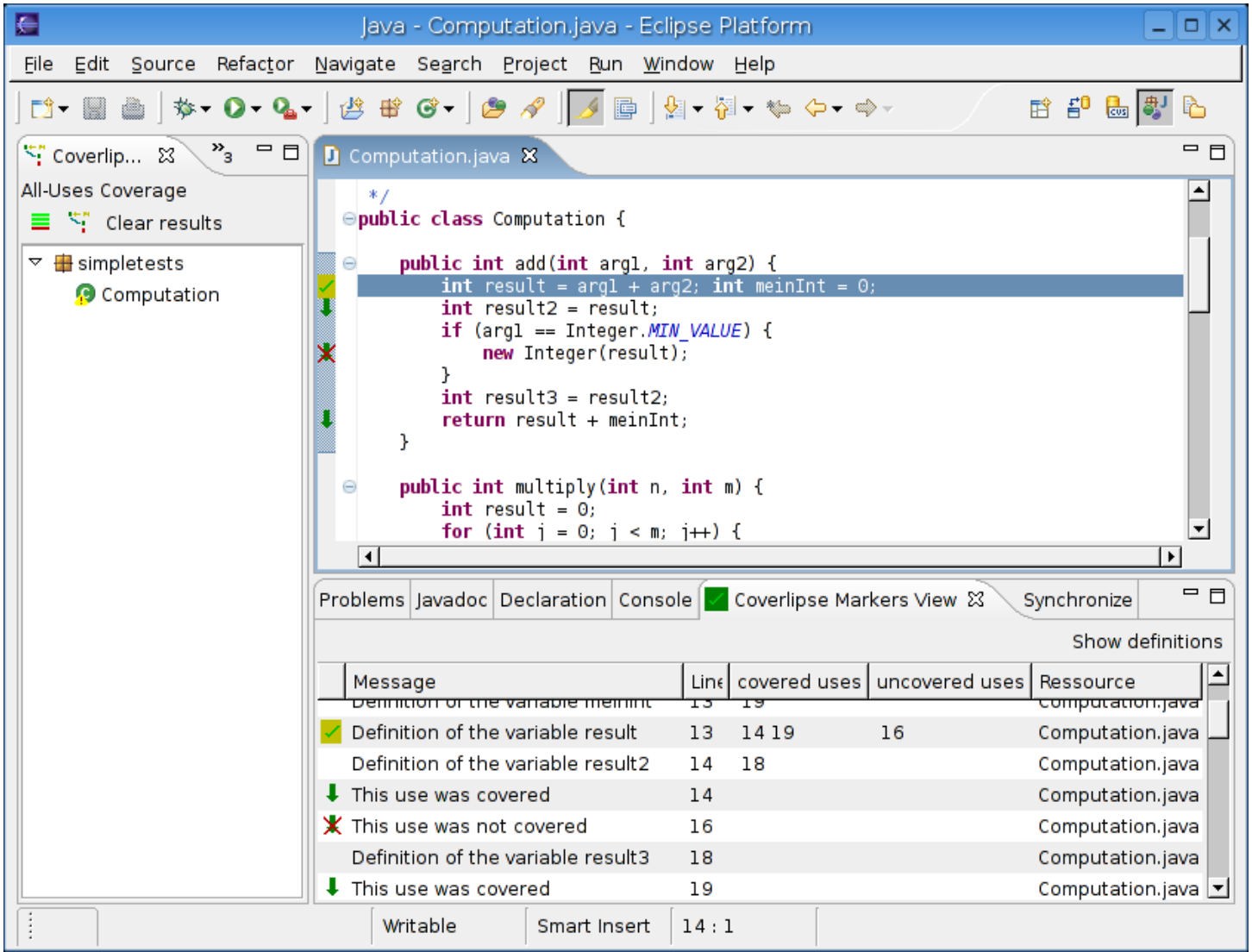
Coverlipse: All-uses Data-Flow Coverage



Coverlipse: Problem Description of a Use of a Variable



Coverlipse: all-uses Coverage

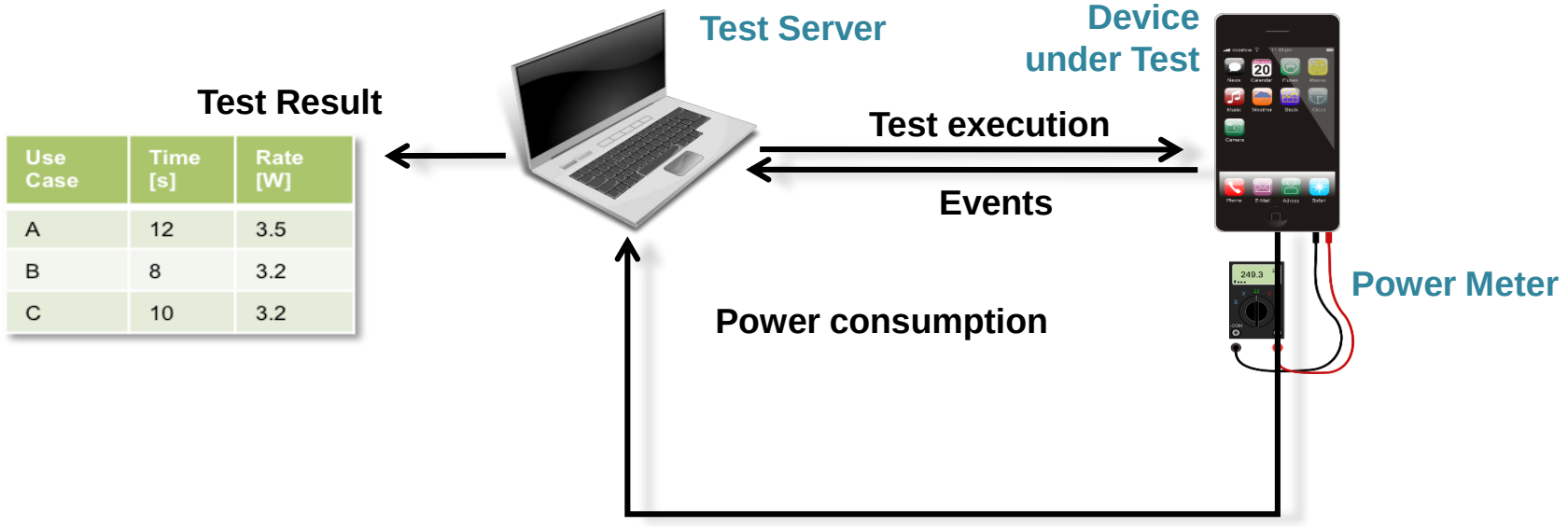


- JouleUnit (courtesy Claas Wilke)
- YouTube-Video: http://is.gd/energyLabel
  http://www.qualitune.org/
  http://www.jouleunit.org/

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



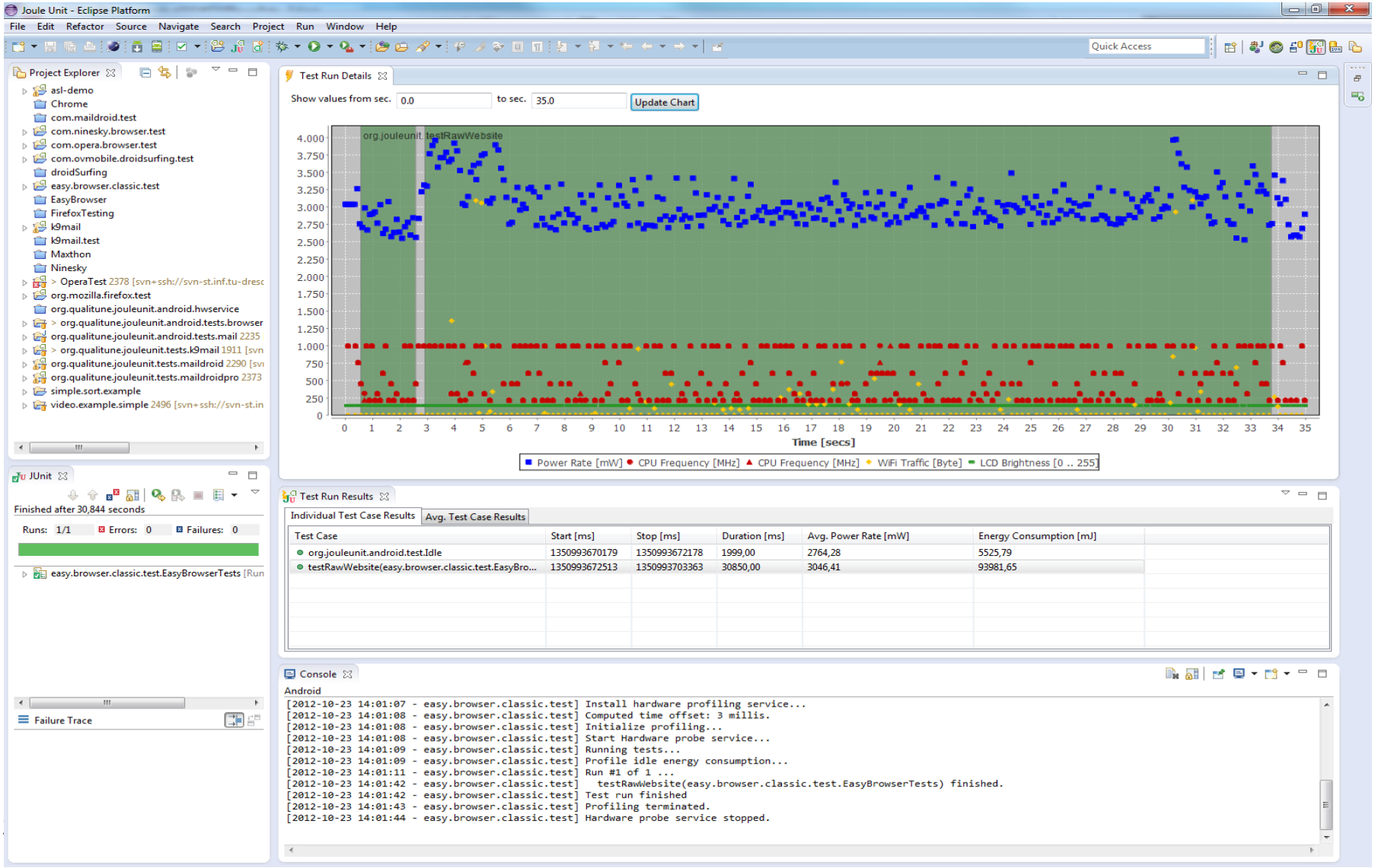
Use Case	Time [s]	Rate [W]
A	12	3.5
B	8	3.2
C	10	3.2

Energy Test

JouleUnit + QMark

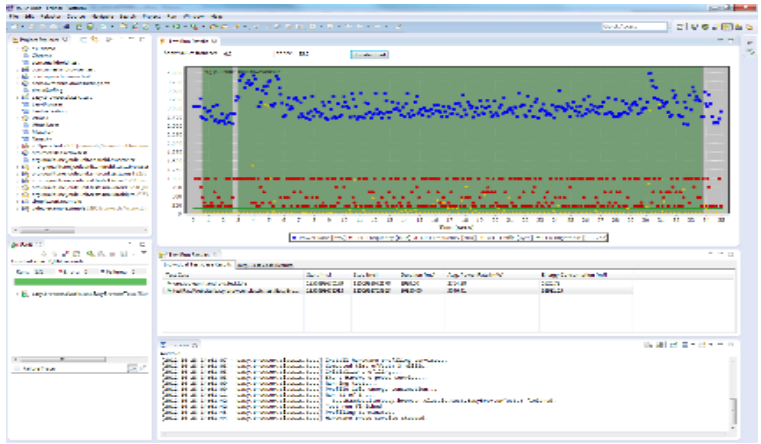
- Extension of JUnit for Energy tests of Android apps
- Reuse of Junit functional tests
- Execution on Smart Phone
  - Feedback on energy bugs
- Remote execution on Qmark test server
  - Hardware energy profiling

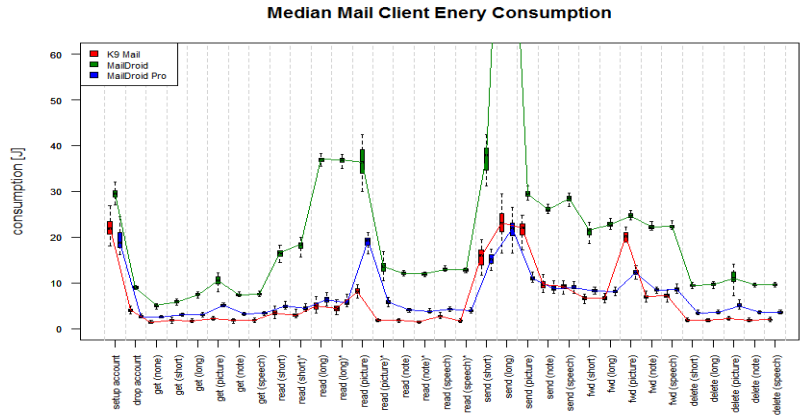
JouleUnit Workbench (Eclipse)

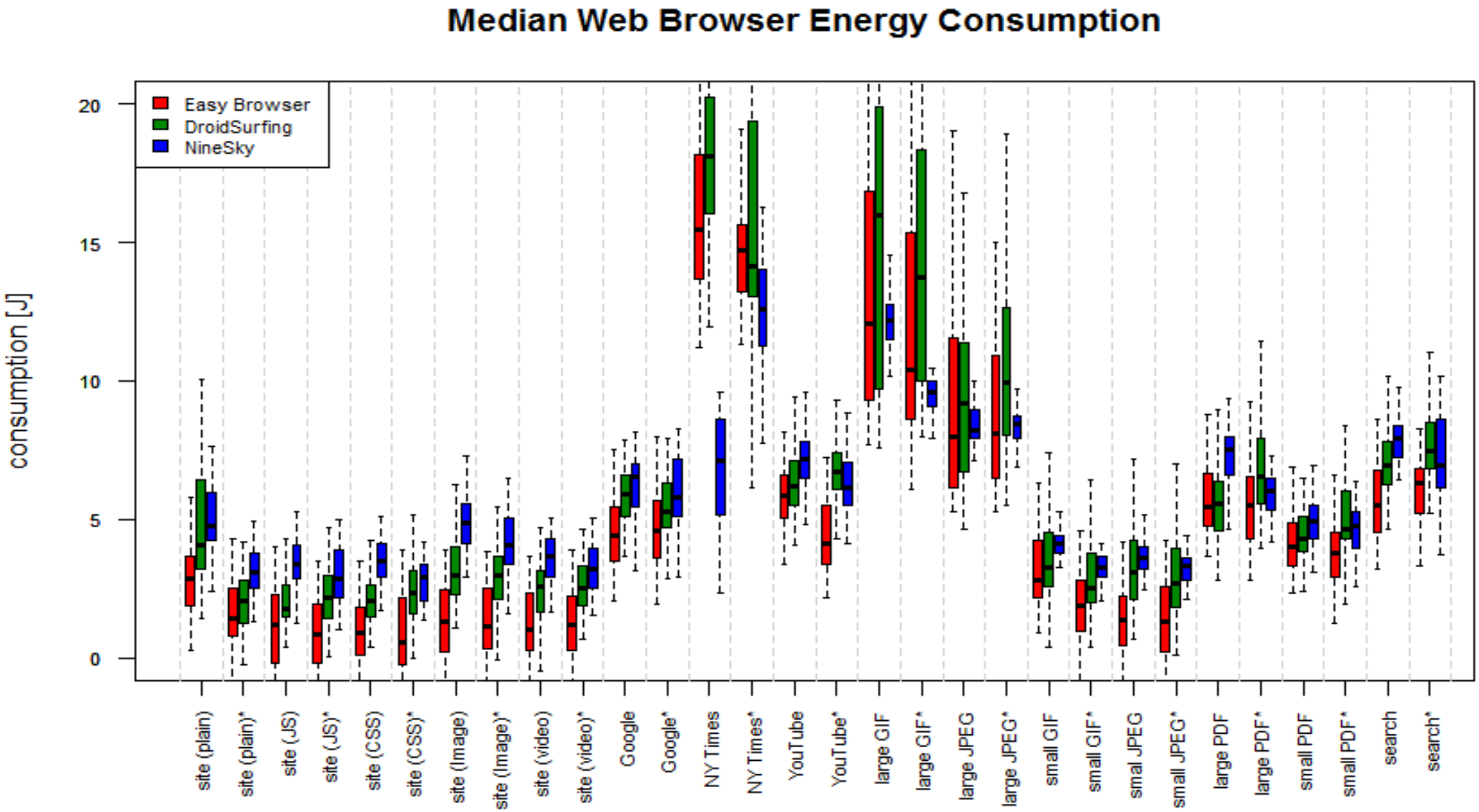


Compare „Simliar“ Apps

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

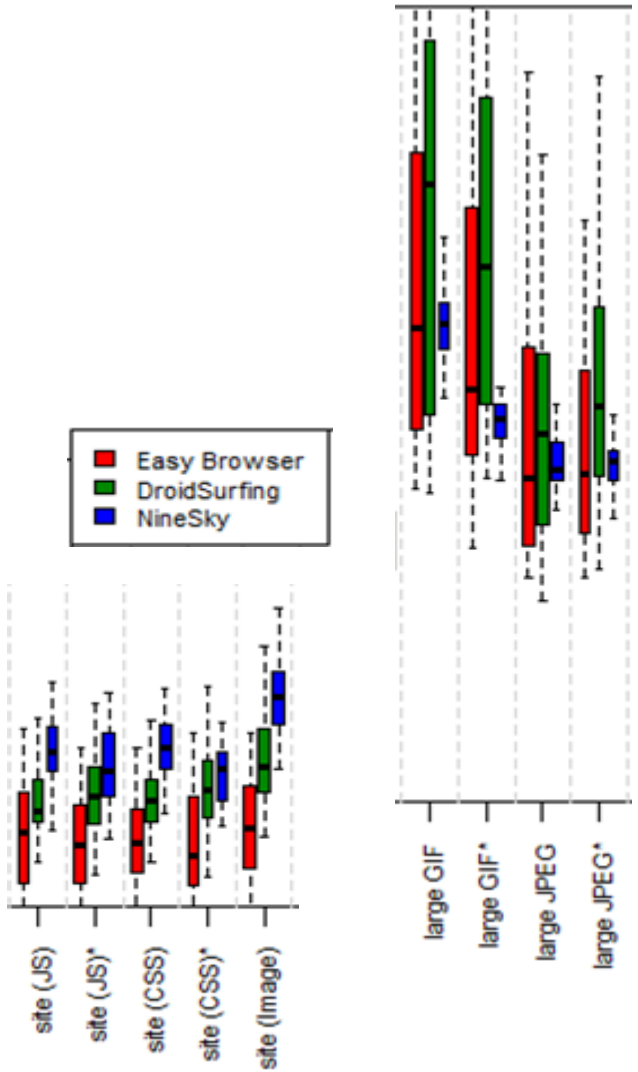


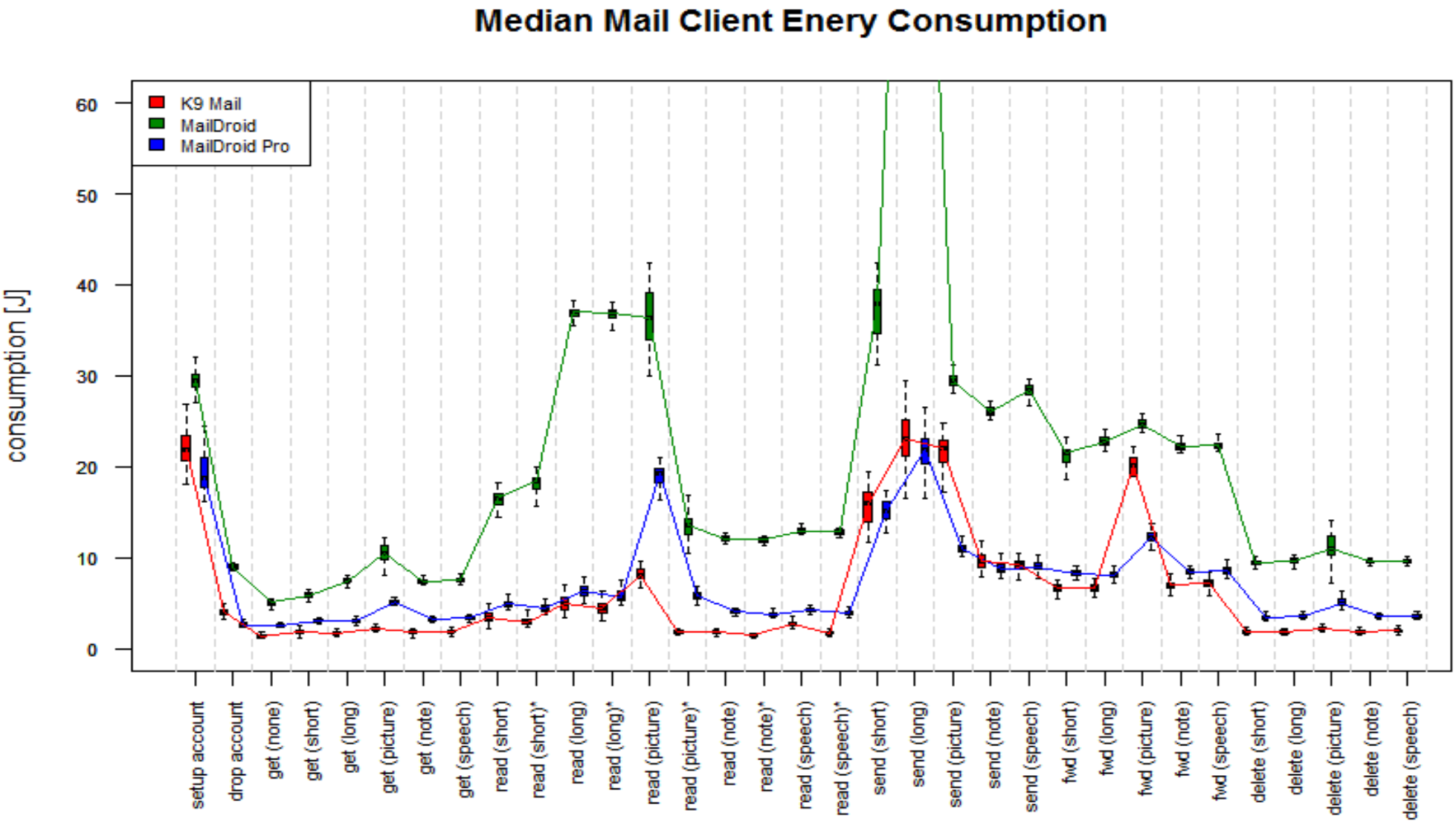



Median Web Browser Energy Consumption
Easy Browser
DroidSurfing
NineSky
consumption [J]
0
5
10
15
20
site (plain)
site (plain)*
site (JS)
site (JS)*
site (CSS)
site (CSS)*
site (Image)
site (Image)*
site (video)
site (video)*
Google
Google*
NY Times
NY Times*
YouTube
YouTube*
large GIF
large GIF*
large JPEG
large JPEG*
small GIF
small GIF*
smal JPEG
small JPEG*
large PDF
large PDF*
small PDF
small PDF*
search
search*

Interesting Issues with Web Browsers

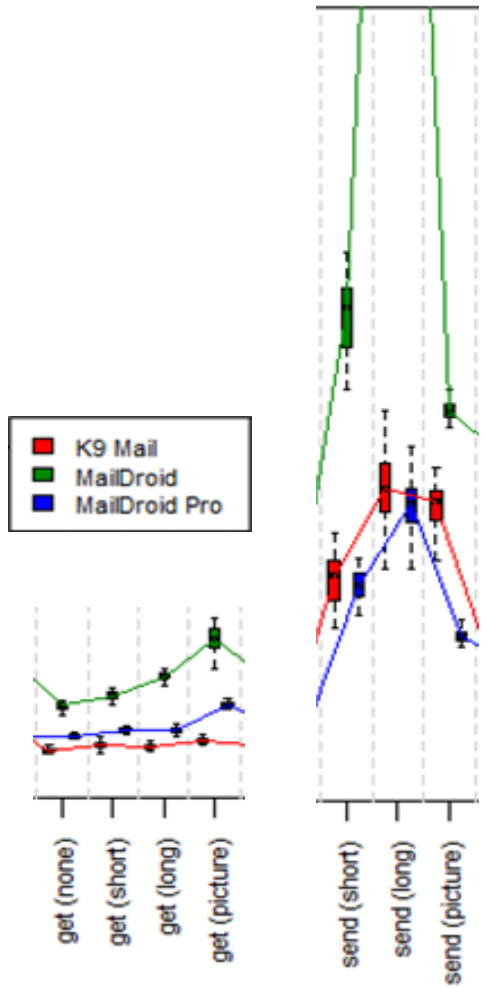
- High variance in measurements (by high variance of response times)
  - But: comparable trends
- NineSky worst for small pages
  - But better for big images (because faster)
- Advertisement in EasyBrowser, DroidSurfing has negative influence only during long load times
- Different browsers are optimal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




Median Mail Client Enery Consumption
K9 Mail
MailDroid
MailDroid Pro
consumption [J]
0
10
20
30
40
50
60
setup account
drop account
get (none)
get (short)
get (long)
get (picture)
get (note)
get (speech)
read (short)
read (short)*
read (long)
read (long)*
read (picture)
read (picture)*
read (note)
read (note)*
read (speech)
read (speech)*
send (short)
send (long)
send (picture)
send (note)
send (speech)
fwd (short)
fwd (long)
fwd (picture)
fwd (note)
fwd (speech)
delete (short)
delete (long)
delete (picture)
delete (note)
delete (speech)

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

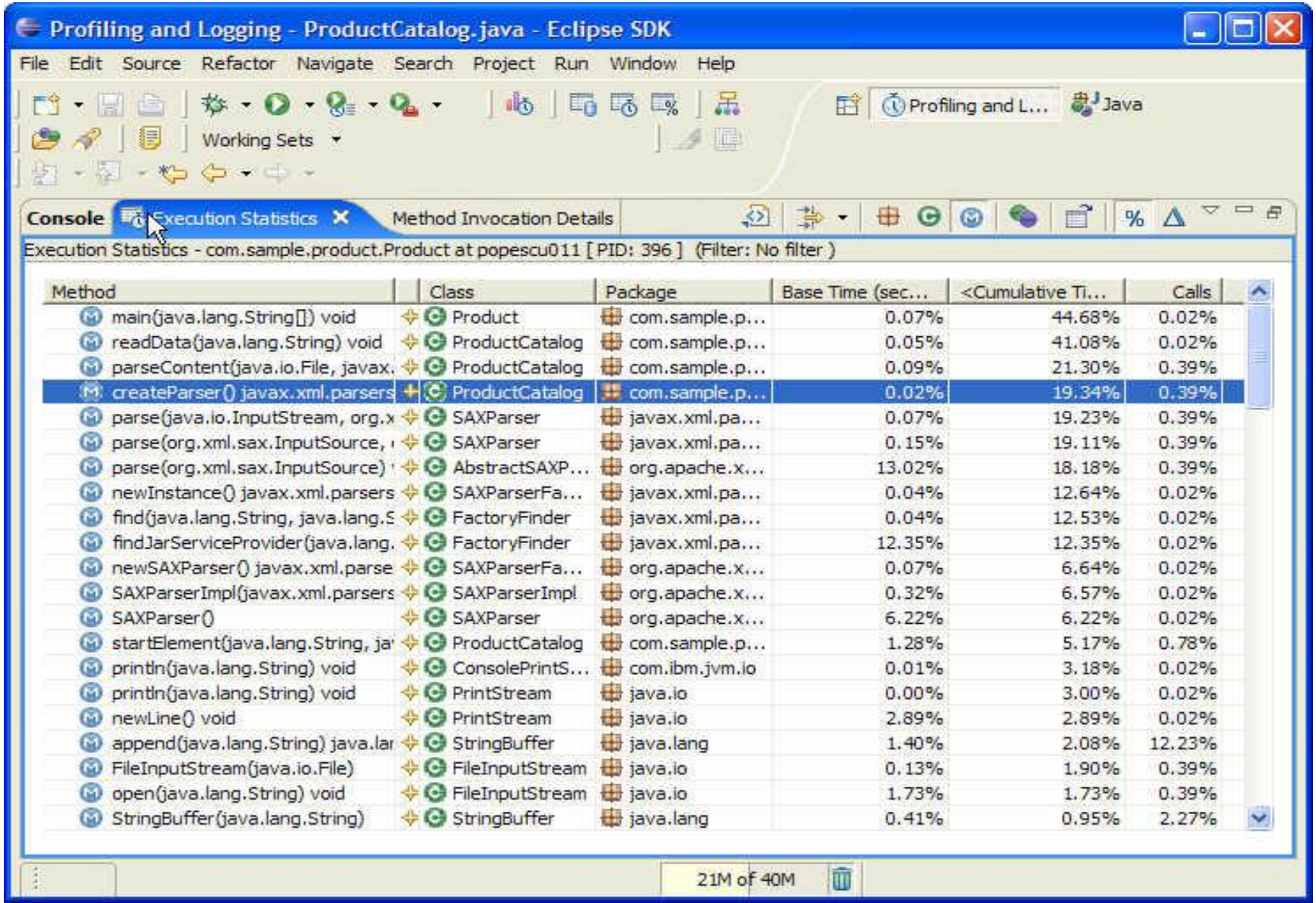
Eclipse Supports Many Test Platform Plugins

- 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

- 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



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

https://confluence.atlassian.com/display/CLOVER/Comparison+of+code+coverage+tools

- 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

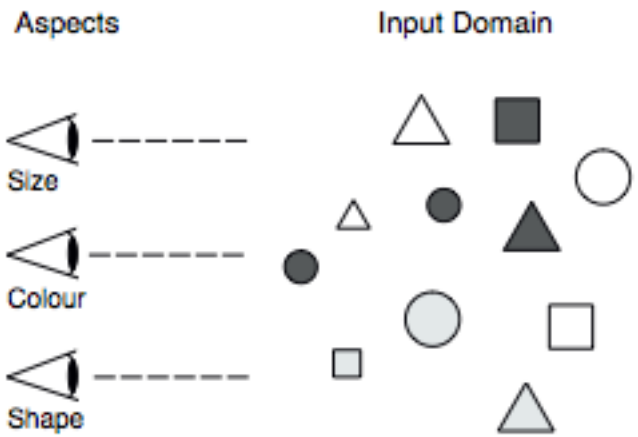
	Enterprise	URL
SilkTest	Segue Software	www.segue.com
TestBench	Imbus	www.imbus.de
Visual 2000	McCabe & Associates, USA	www.mccabe.com
Cantata++	IPL, Bath, UK	www.iplbath.com
ClickTracks	ClickTracks Analytics, Inc.CA	www.clicktracks.com
Tracetronic ECUTest	Tracetronic, Dresden. For motor tests	www.tracetronic.de

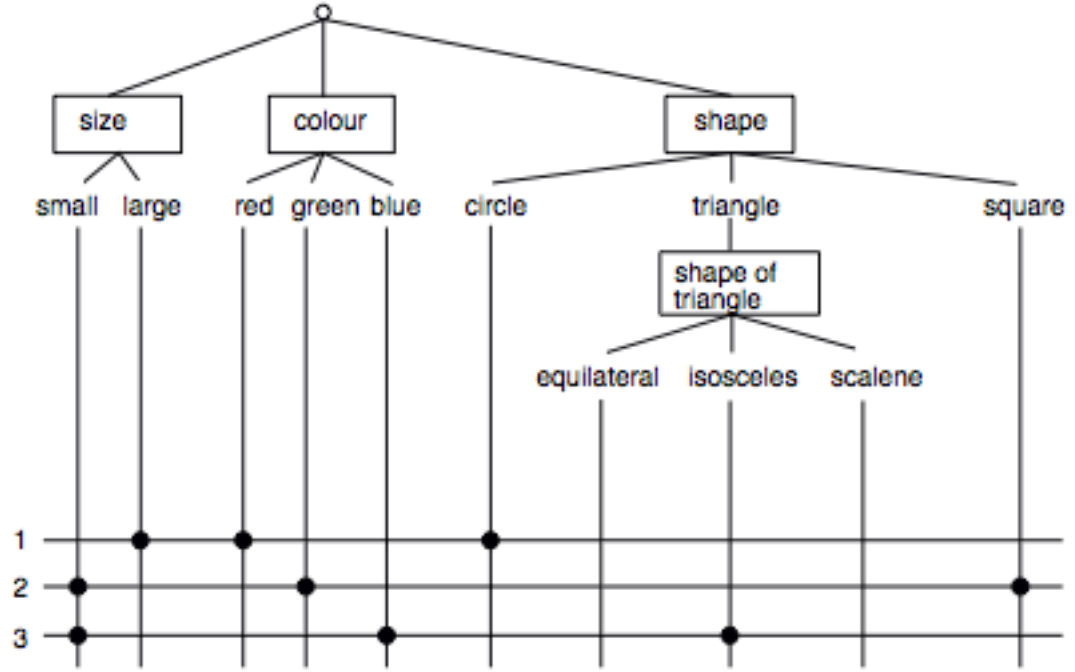
More: https://www.testtoolreview.de/de/

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*



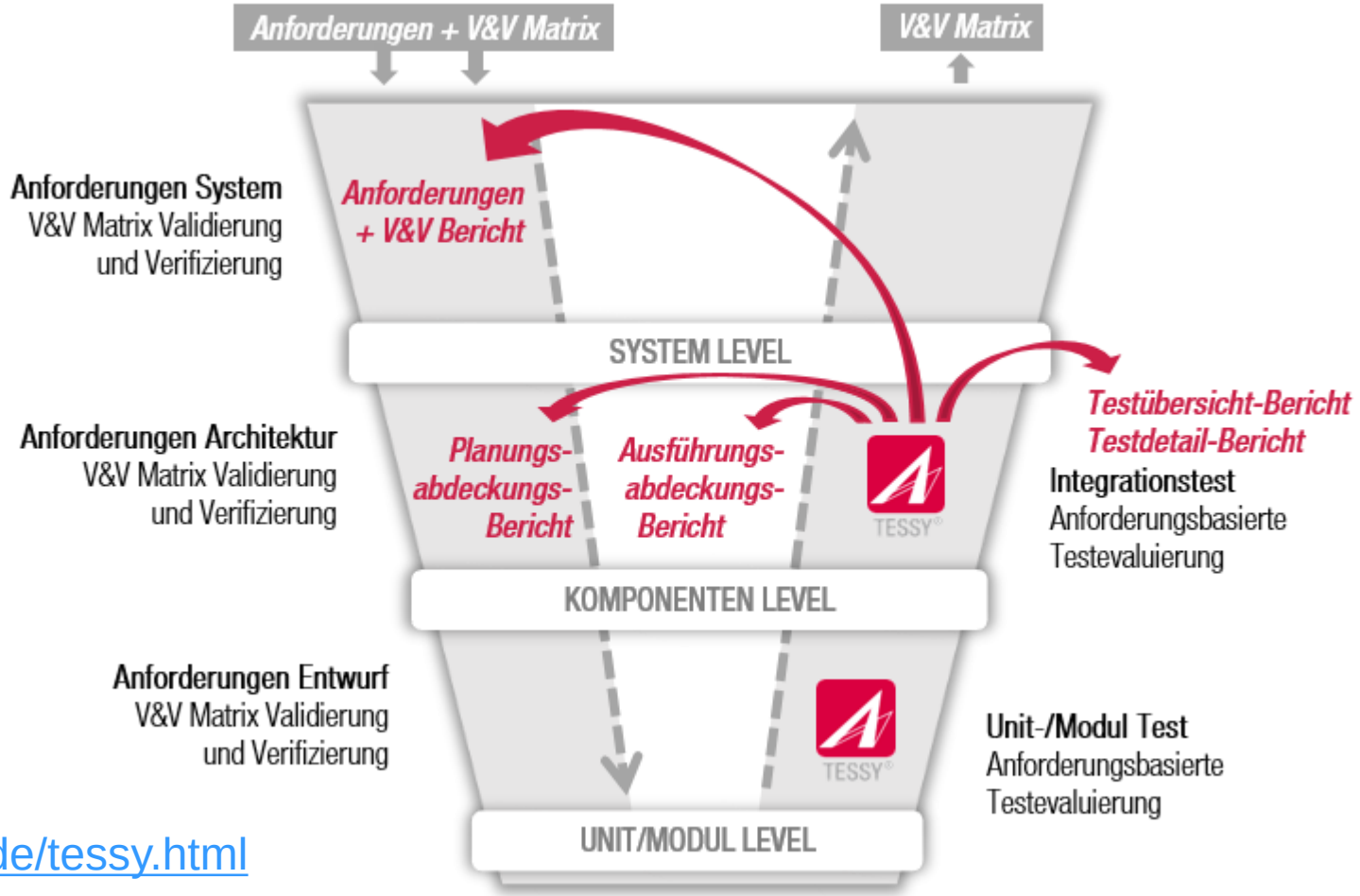


Advantages of Classification Tree Method

- 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 choosen as boundary values of the intervals
- Coverage
  - A good combination of parameter values should cover the most important error cases
  - Generation of test cases

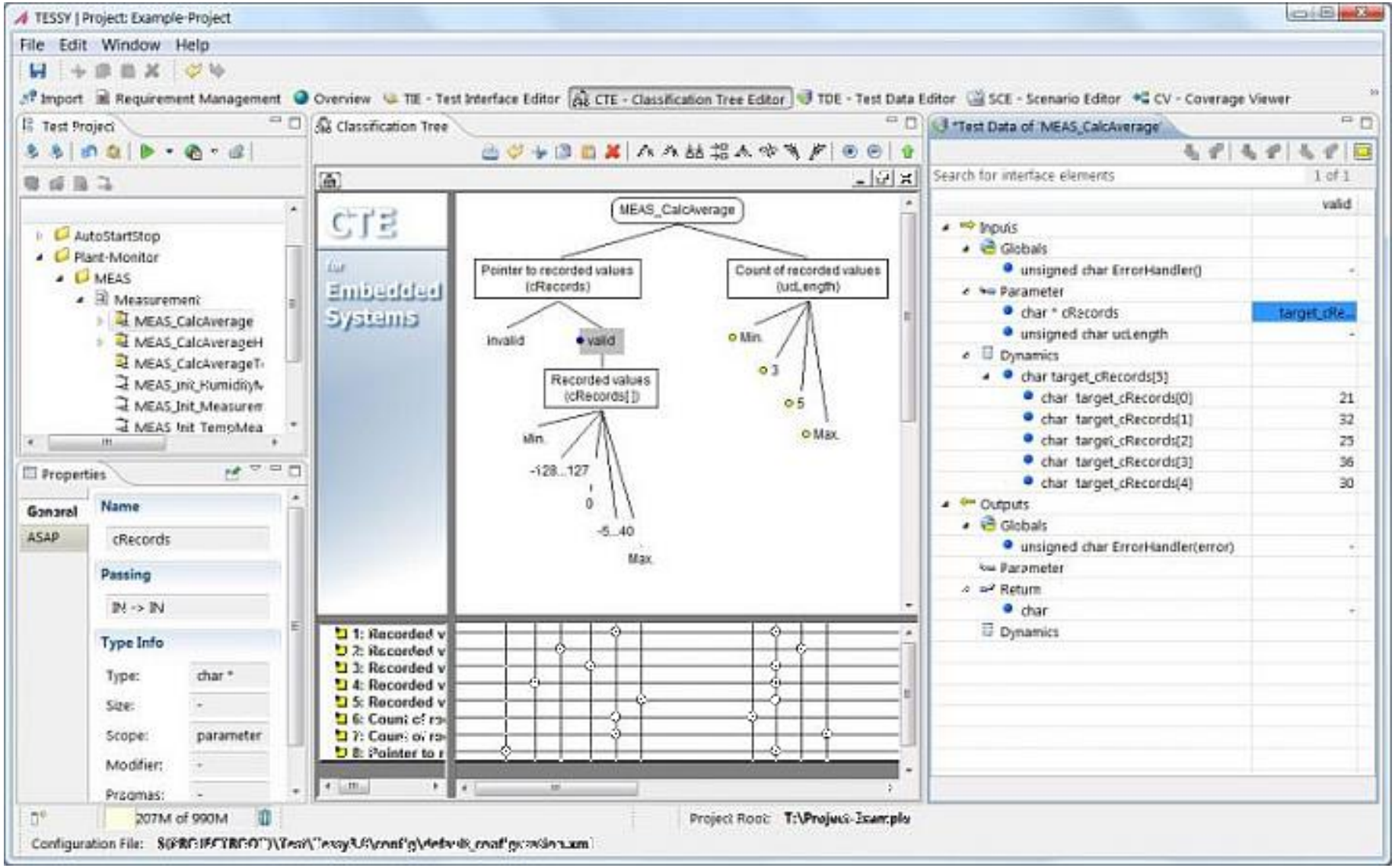
TESSY of HITEX

- 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



© Prof. U. Aßmann

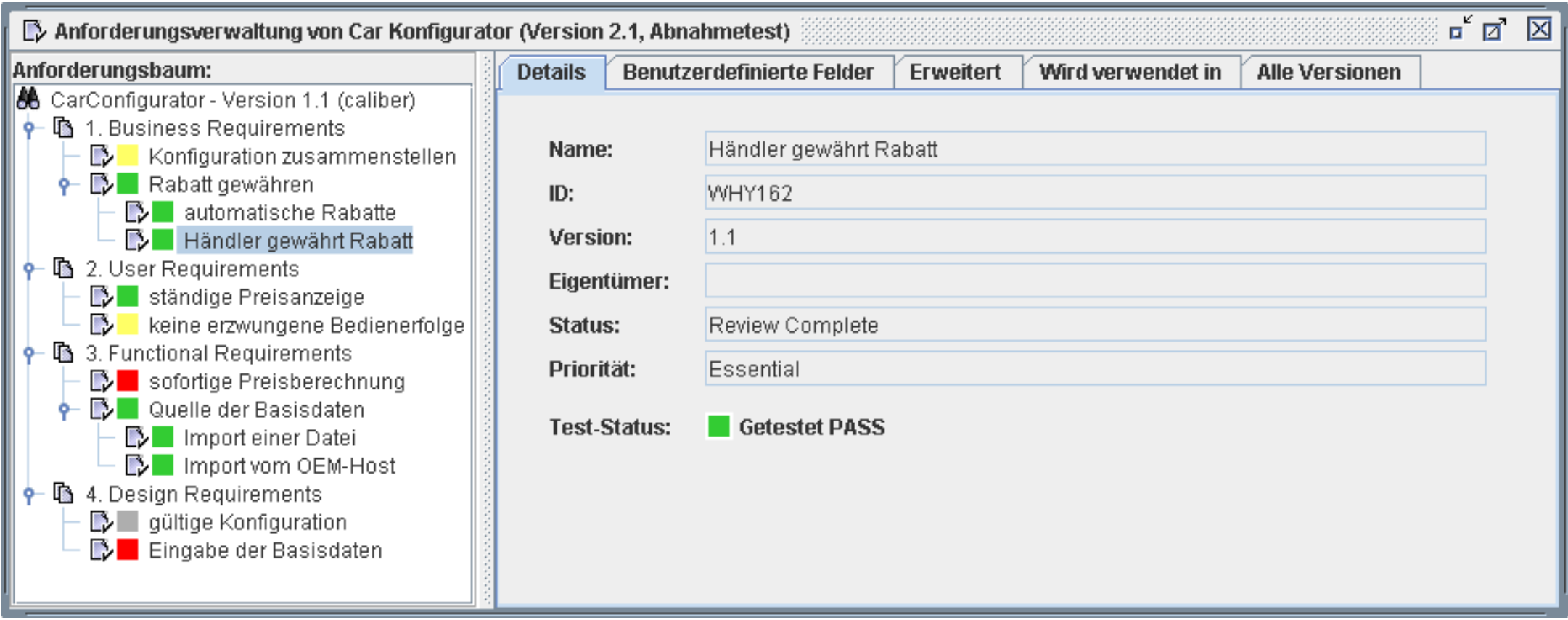
12.3.2 IMBUS TESTBENCH

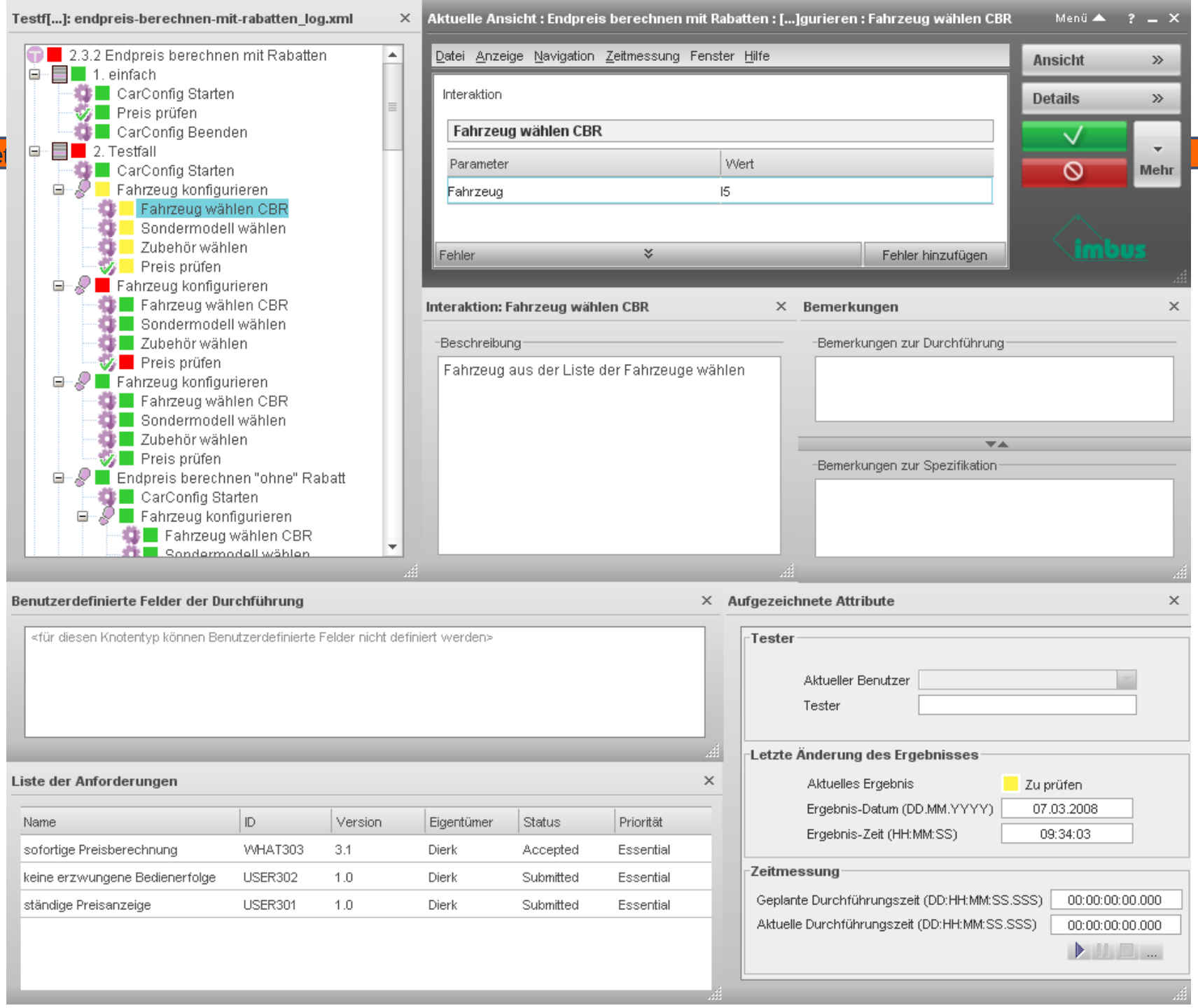
imbus TestBench

- 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)



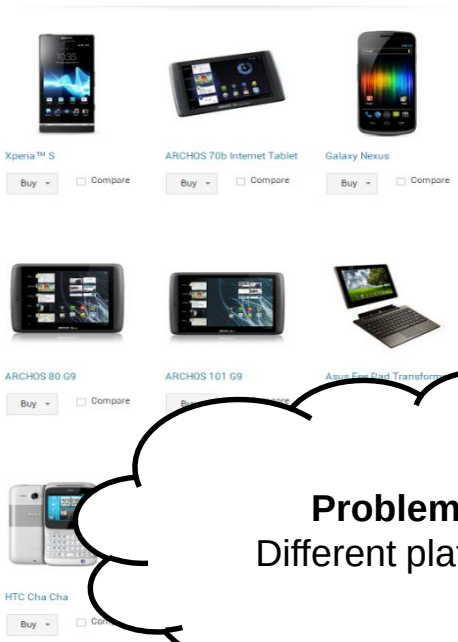

Testf[...]: endpreis-berechnen-mit-rabatten_log.xml
2.3.2 Endpreis berechnen mit Rabatten
1. einfach
CarConfig Starten
Preis prüfen
CarConfig Beenden
2. Testfall
CarConfig Starten
Fahrzeug konfigurieren
Fahrzeug wählen CBR
Sondermodell wählen
Zubehör wählen
Preis prüfen
Fahrzeug konfigurieren
Fahrzeug 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
Aktuelle Ansicht : Endpreis berechnen mit Rabatten : [...]gurieren : Fahrzeug wählen CBR
Menü
Datei Anzeige Navigation Zeitmessung Fenster Hilfe
Interaktion
Fahrzeug wählen CBR
Parameter
Wert
Fahrzeug
I5
Fehler
Fehler hinzufügen
Ansicht
Details
Mehr
imbus
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)
07.03.2008
Ergebnis-Zeit (HH:MM:SS)
09:34:03
Zeitmessung
Geplante Durchführungszeit (DD:HH:MM:SS.SSS)
00:00:00:00.000
Aktuelle Durchführungszeit (DD:HH:MM:SS.SSS)
00:00:00:00.000
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

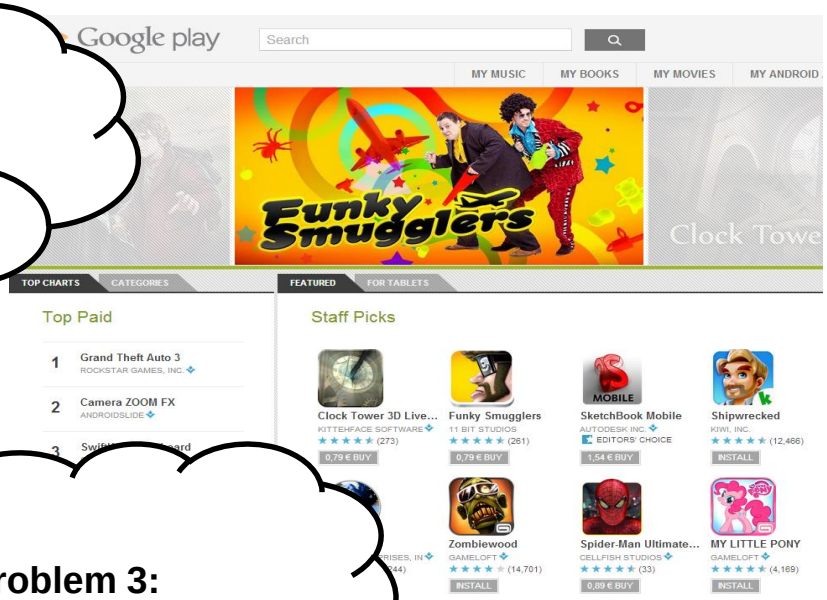
Model-Driven Testing

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



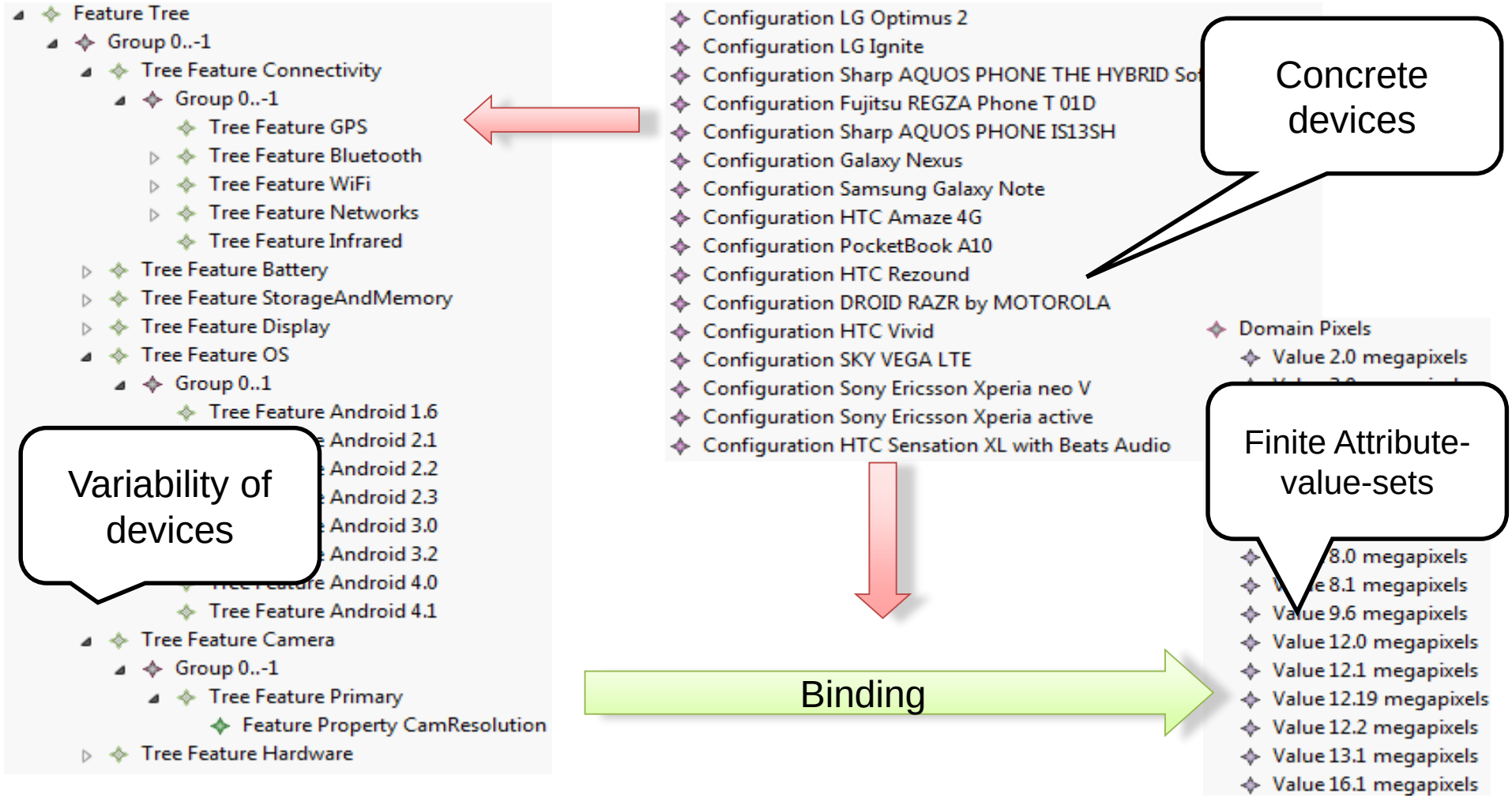
Problem 1:
Different platforms

Problem 2:
Apps differ on different devices



Problem 3:
Apps are context-adaptive

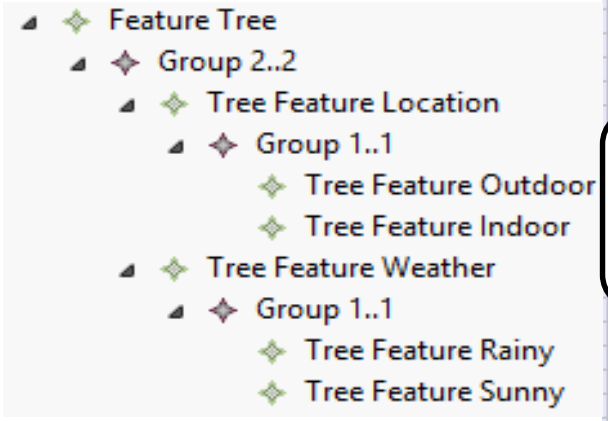
Plattform Management with Attributed Feature-Models (And-Or-Trees)

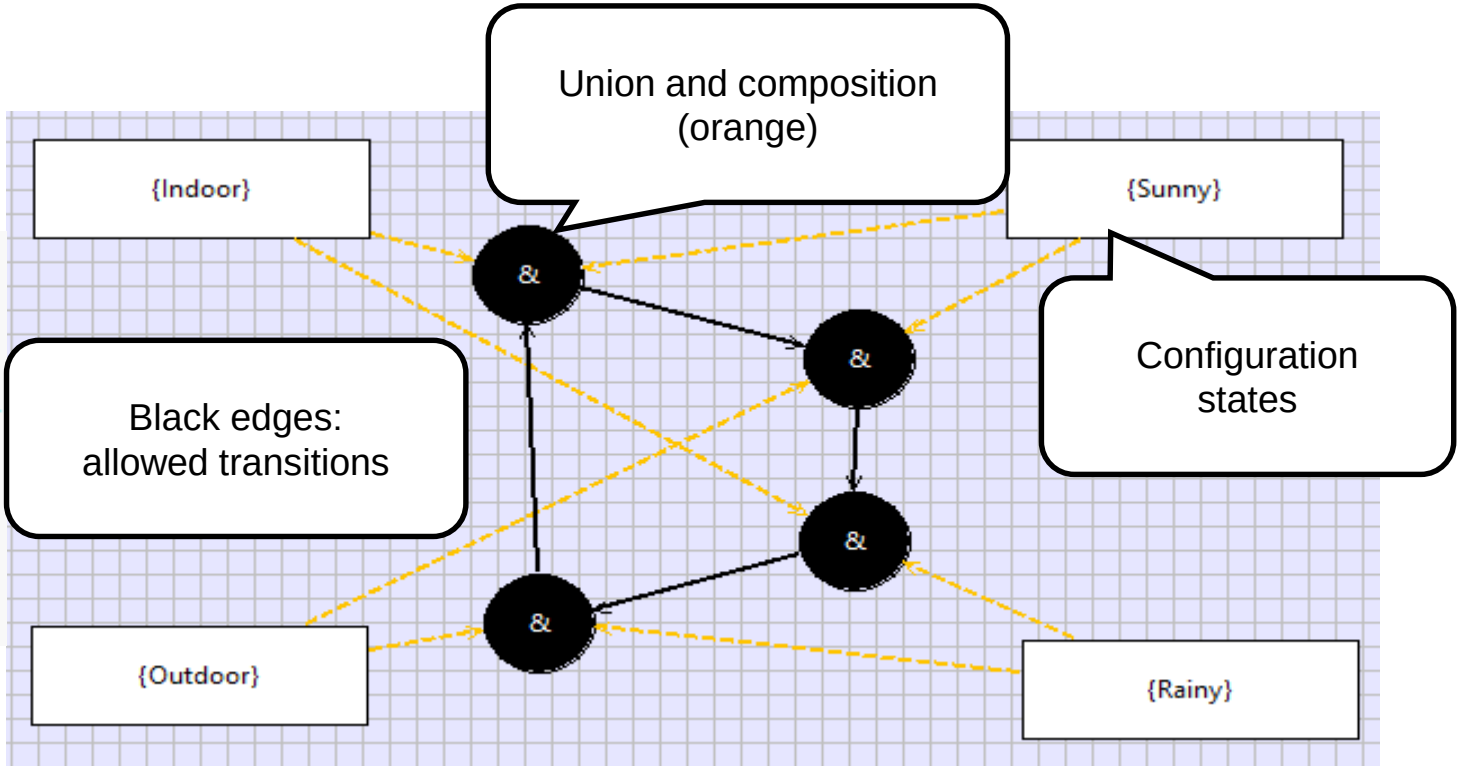


Variability in applications can be described by feature models

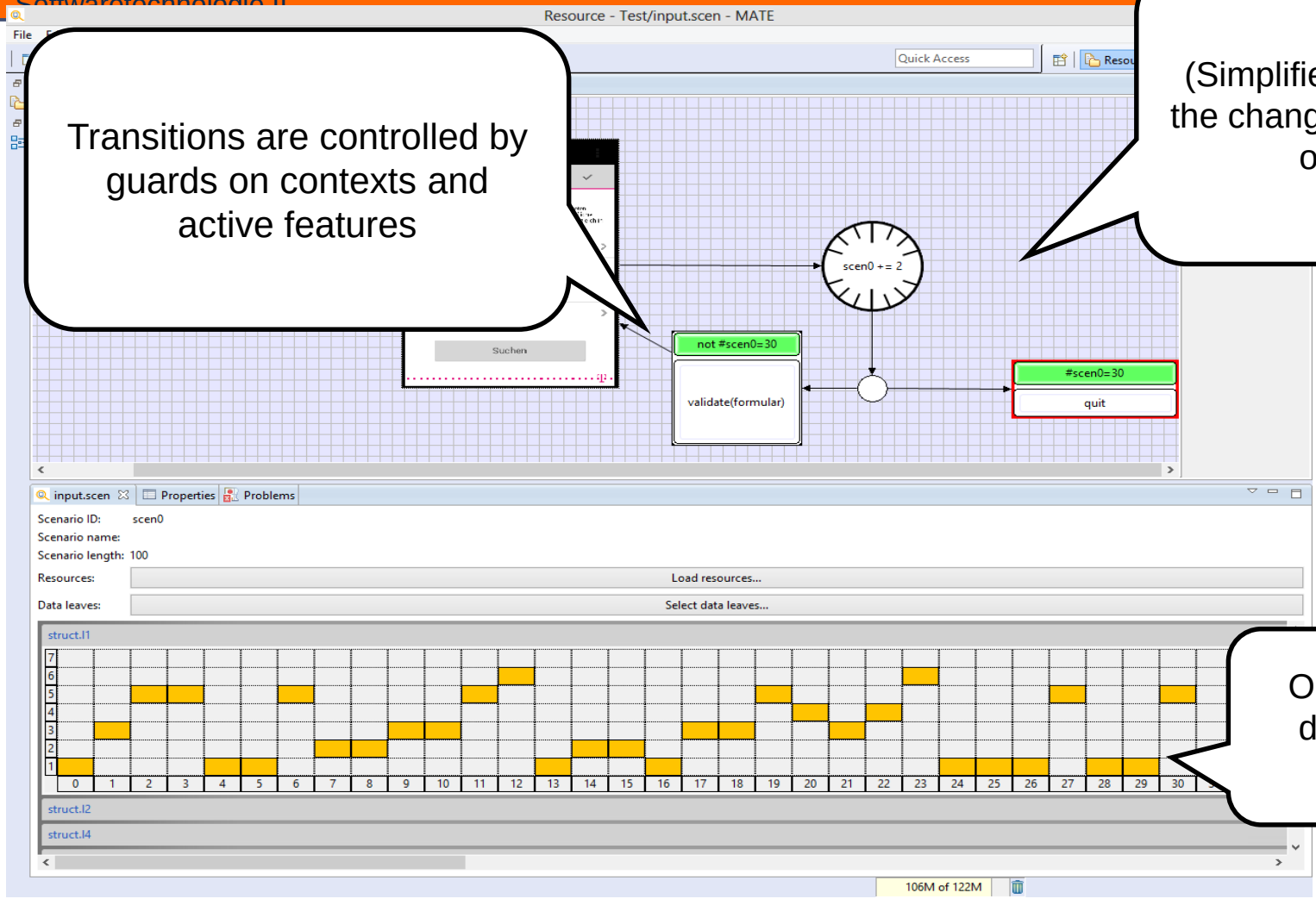
Dynamic Change of Features

- 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



Transitions are controlled by guards on contexts and active features

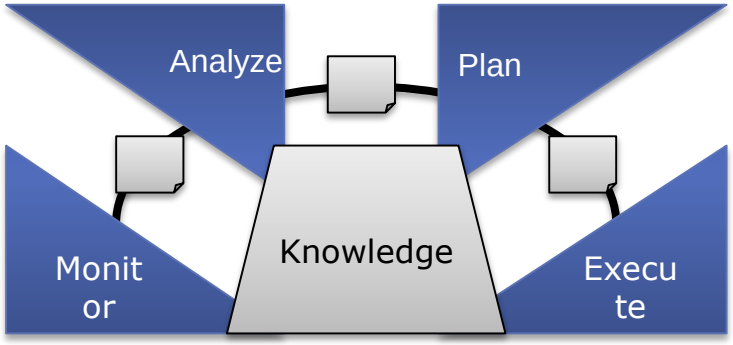
(Simplified) Timed Petrinet controls the changes of contexts in Scenarios or OCM alternatives

OCM alternative: Szenarios describe value changes of context data

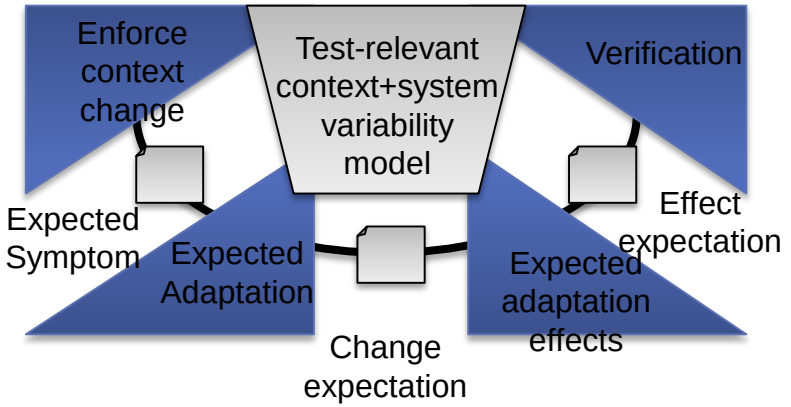
The MATE generator produces test cases by reachability analysis on the timed petrinet.

Test of SAS must be Model-Based

- Self-adaptive Systems (SAS) reconfigure themselfes
  - 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

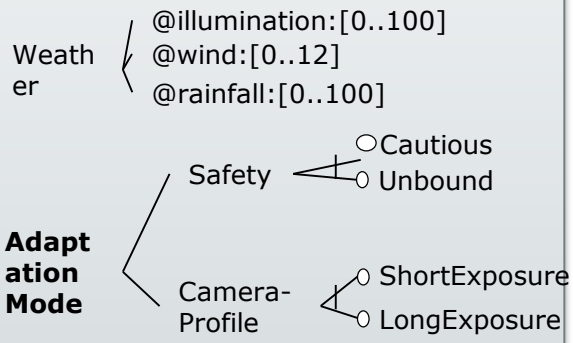
- Solution A: Model-driven Testing (MDT)
  - Black-box testing with automatic test design (in contrast to automatic test exectution)
  - Test data, cases, and expected outcomes are generated from models
  - The models‘ expressiveness hides complexity
  - Adequacy criteria control generation

- 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)

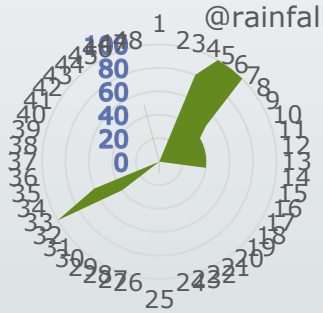
- implements both MDT and ITL for SAS
- provides SAS-specific **test metamodels**, interpretors, tooling, and a test adaptor framework

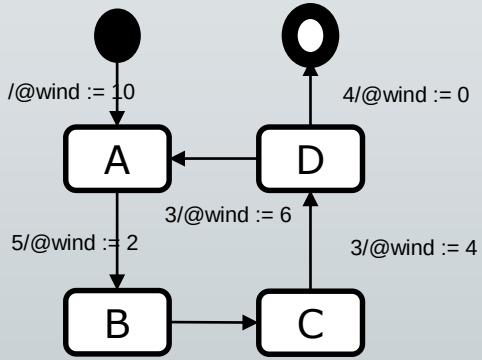
Context & System variability



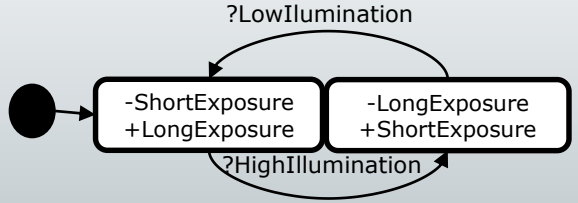
@wind > 4 and @rainfall > 20 ⇒ Cautios
@illumination < 60 ⇒ LongExposure

Context dynamics

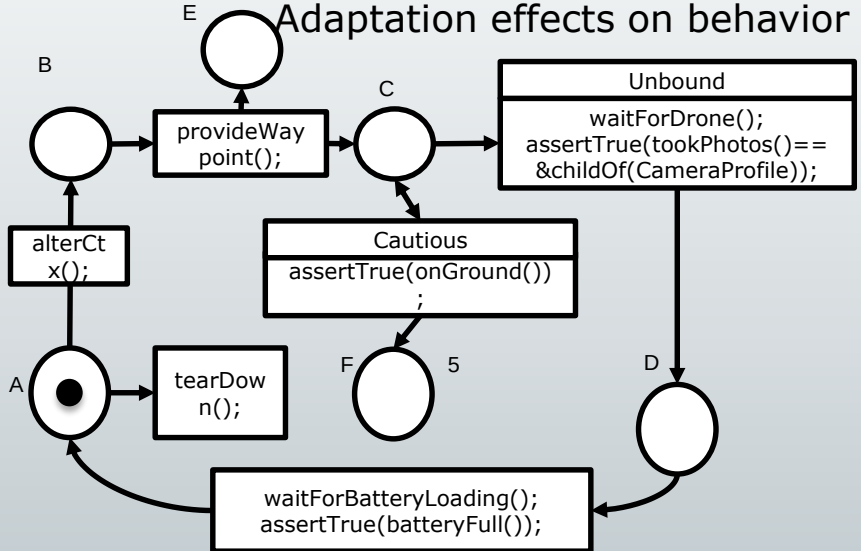




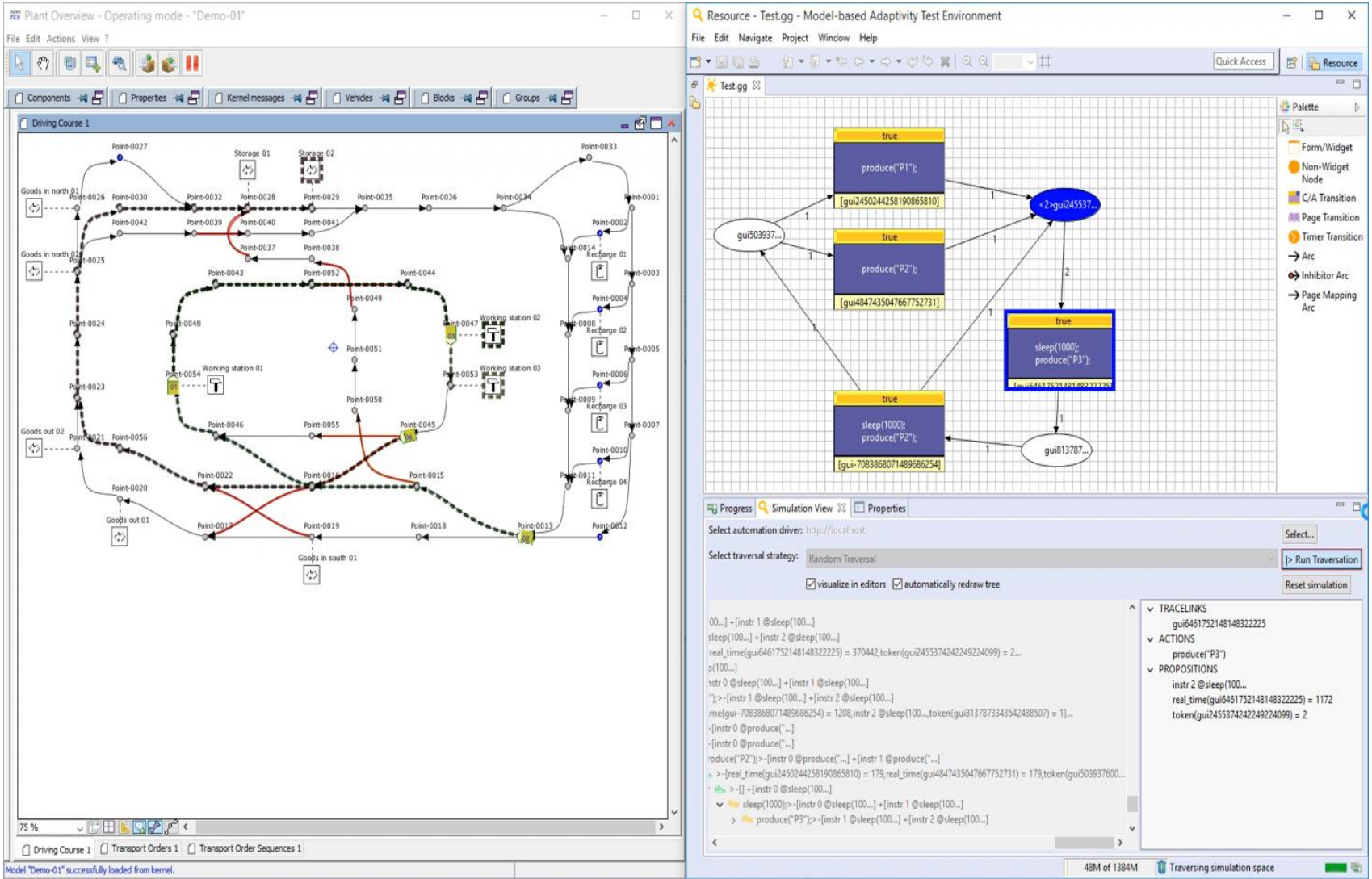
Adaptation dynamics



Adaptation effects on behavior



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



The End