TECHNISCHE UNIVERSITÄT DRESDEN

DRESDEN concept

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 2019, 27.11.2019

12. Validation Tools

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

12.1 Code-Centric Test-Frameworks

12.1.1 Code Coverage Tools

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

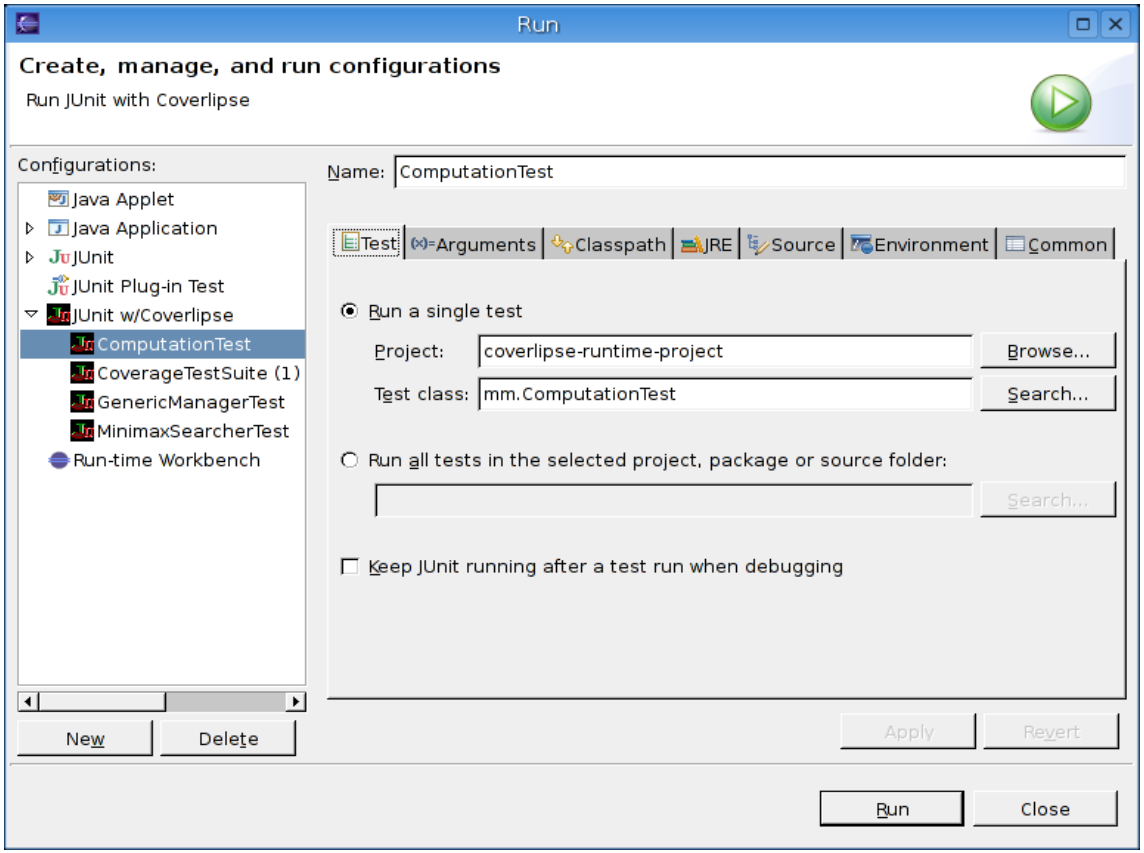
Coverage class	Technique	Purpose
Statement	How many statements are covered by how many test cases?	Discover dead code
Condition (Alternative)	How many branches are covered by how many test cases?	Cover all edges in control-flow graph
Combination of conditions	How many combinations of subsequent branches 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)

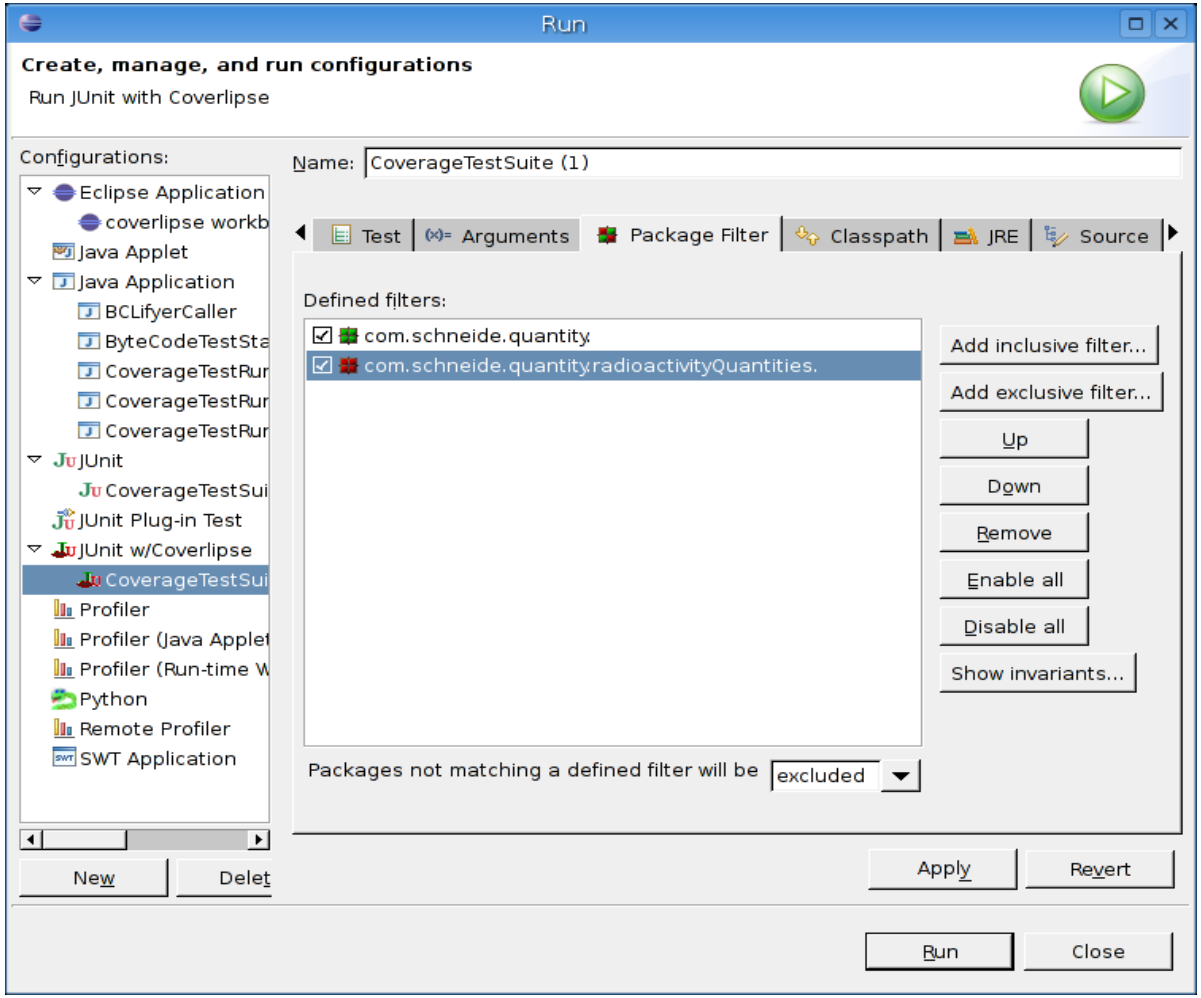
Coverage class	Technique	Purpose
All defs	For all definitions (assignments) of variables: one path to a use has to be tested	Discover dead variable assignments (definitions)
All p-uses	For a definition (assignment) of a variable, test all uses *in predicates*	Show the influence of the variable assignment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

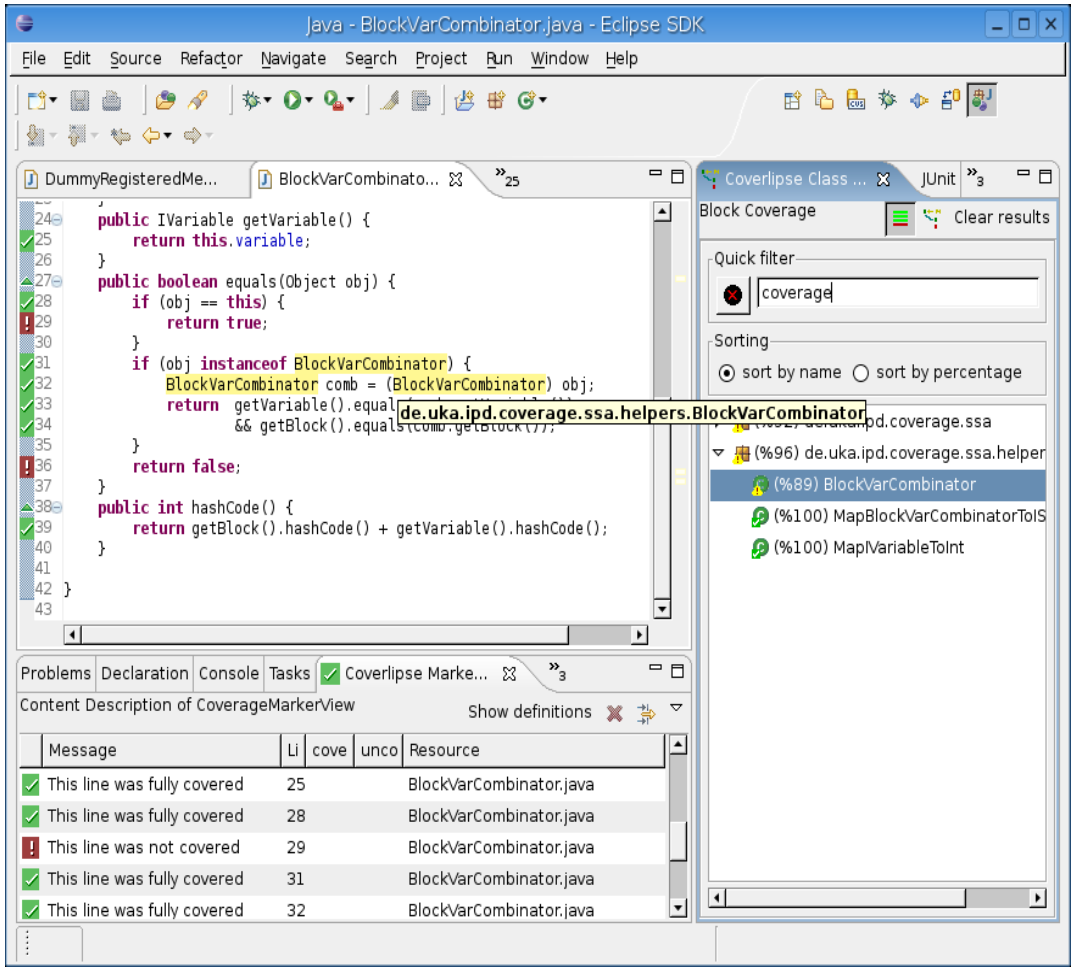
- Selection of Junit test cases and their path coverage analysis



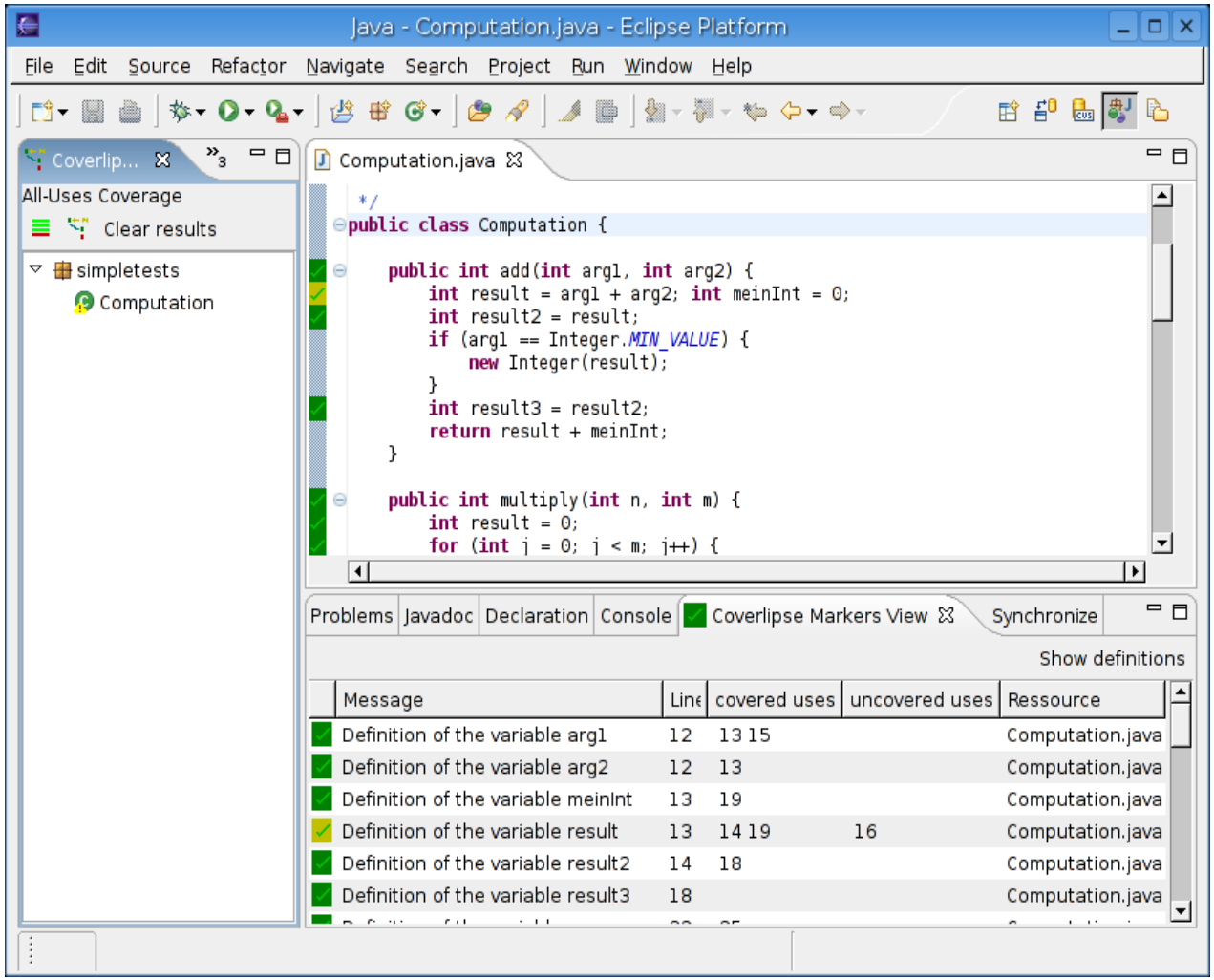
Coverlipse– Selecting Packages to Analyze



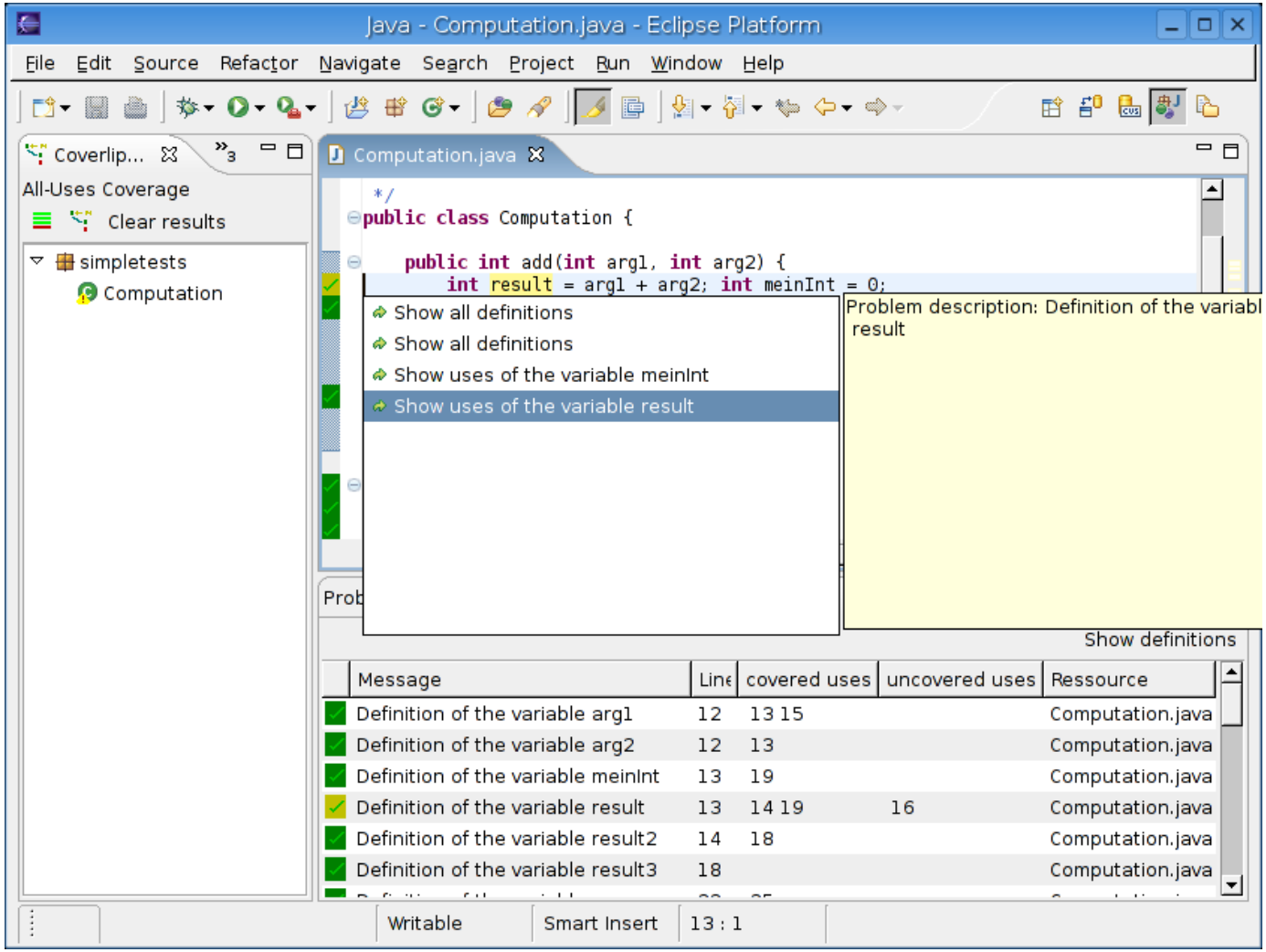
Coverlipse Block Coverage / Statement Coverage



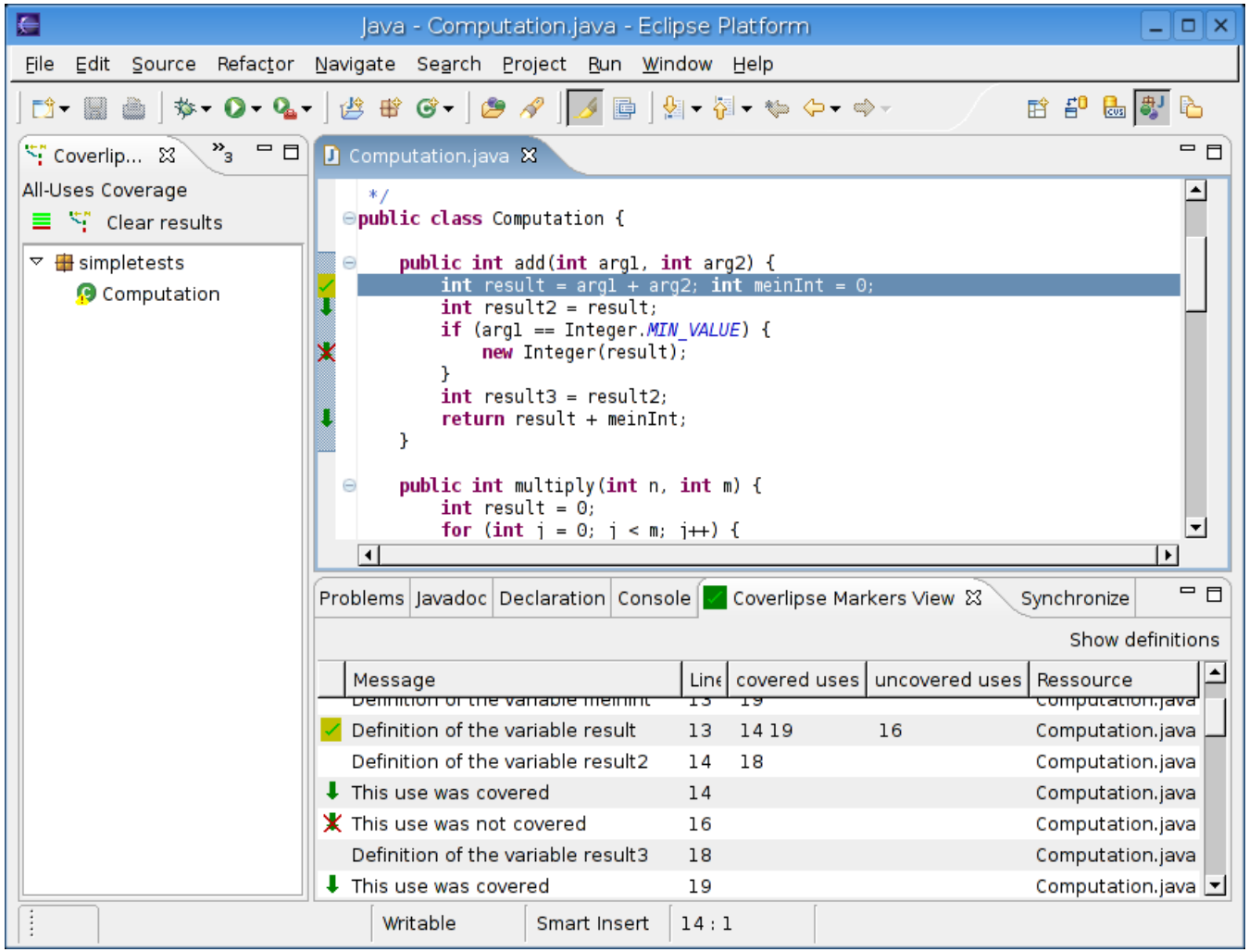
Coverlipse: All-uses Data-Flow Coverage



Coverlipse: Problem Description of a Use of a Variable



Coverlipse: all-uses Coverage

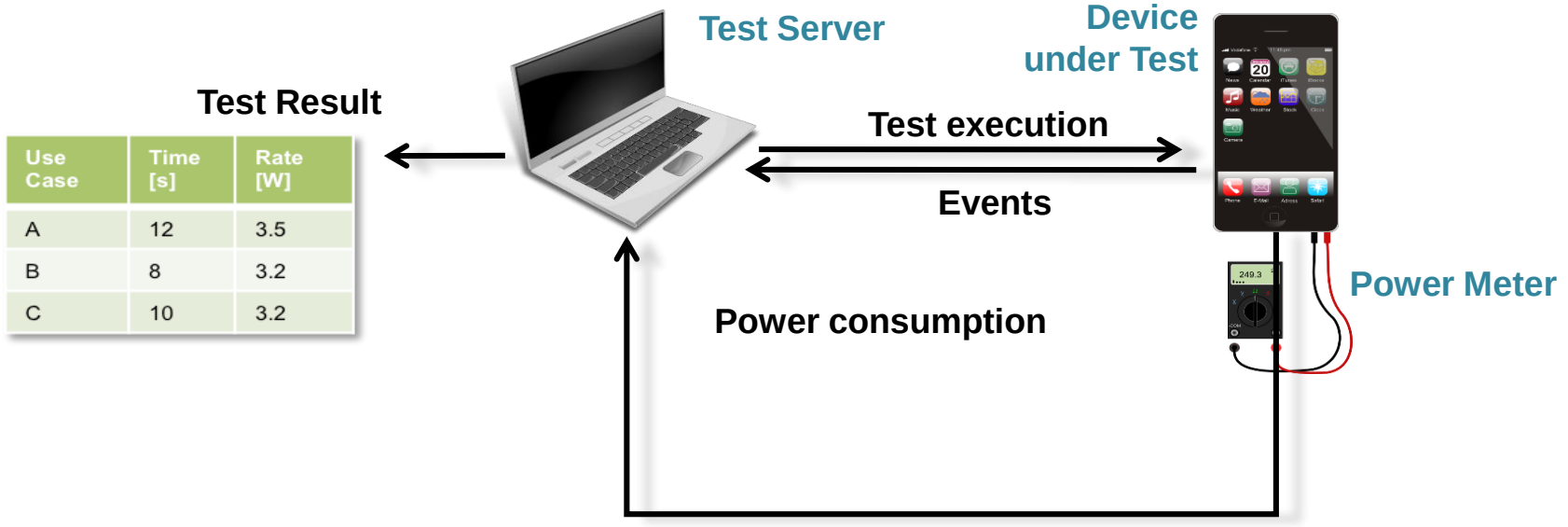


JouleUnit (courtesy Claas Wilke)
YouTube-Video: http://is.gd/energyLabel

12.1.2 Code-Centric Energy Test-Frameworks

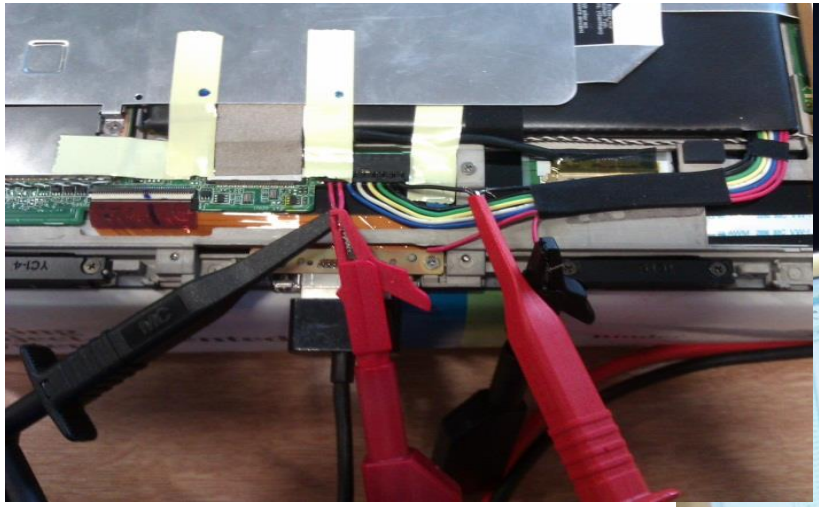
Energy Test with JouleUnit

- 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

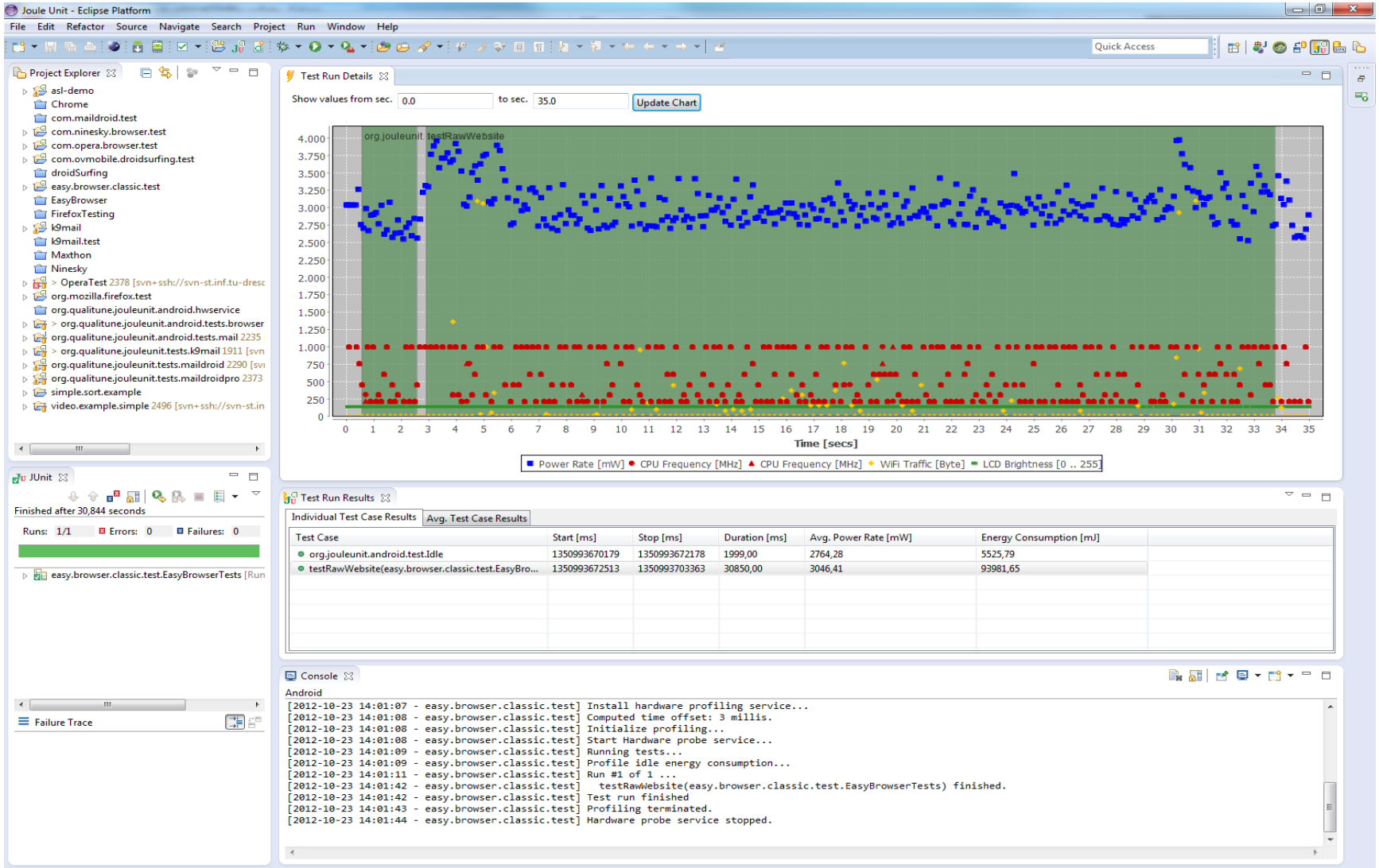
Energy Test



JouleUnit + QMark

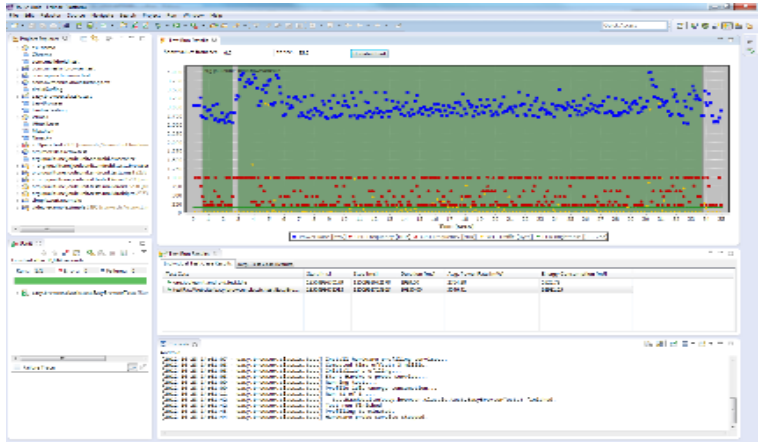
- Extension of JUnit for Energy tests of Android apps
- Reuse of Junit functional tests
- Execution on Smart Phone
  - Feedback on energy bugs
- Remote execution on Qmark test server
  - Hardware energy profiling

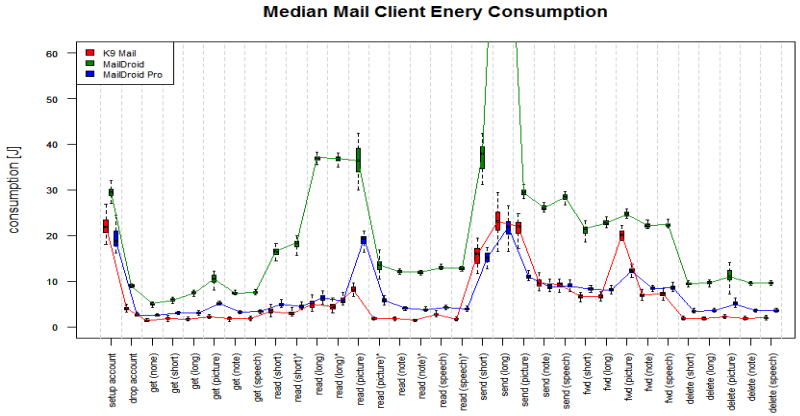
JouleUnit Workbench (Eclipse)



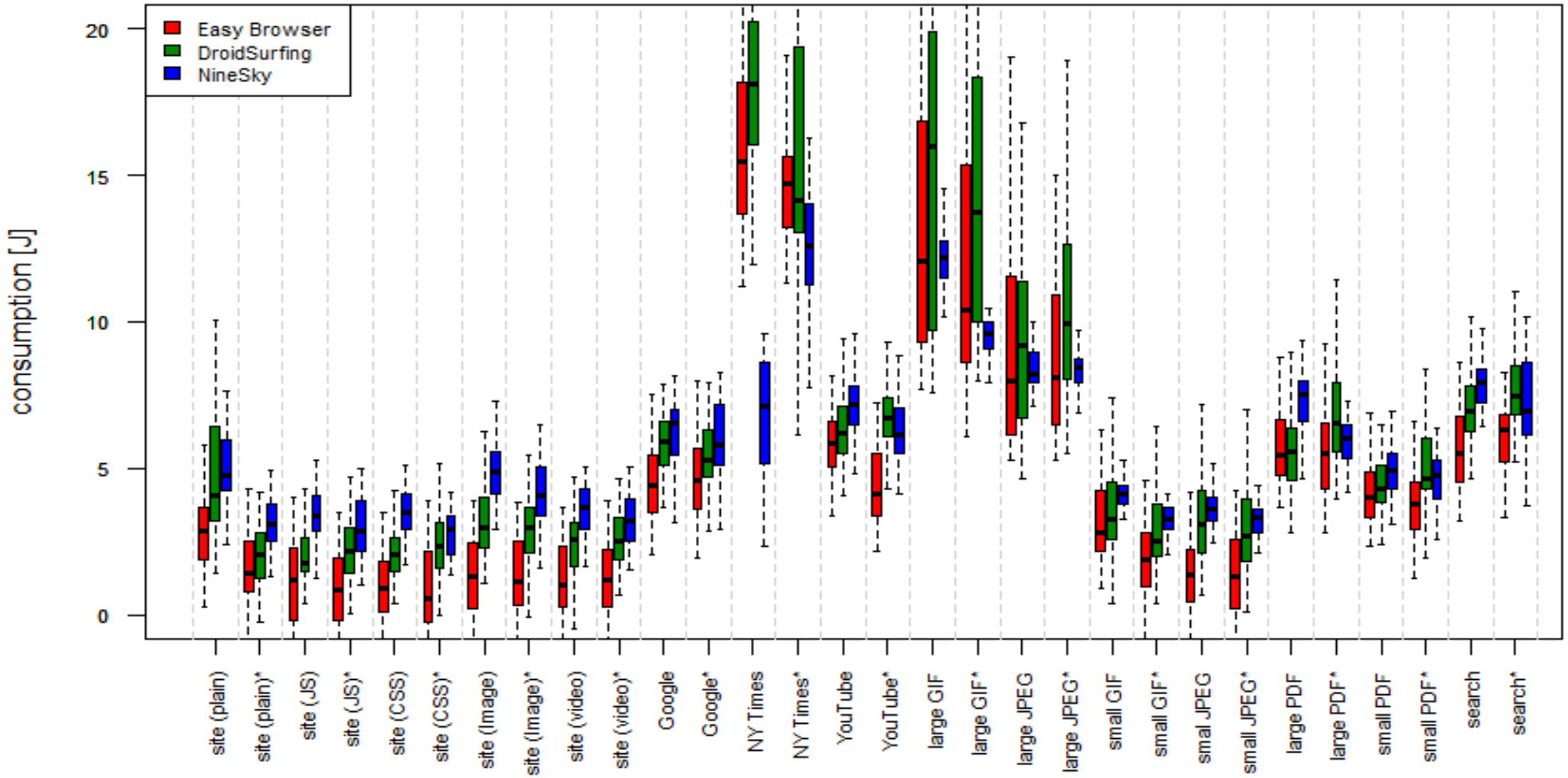
Compare „Simliar“ Apps

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



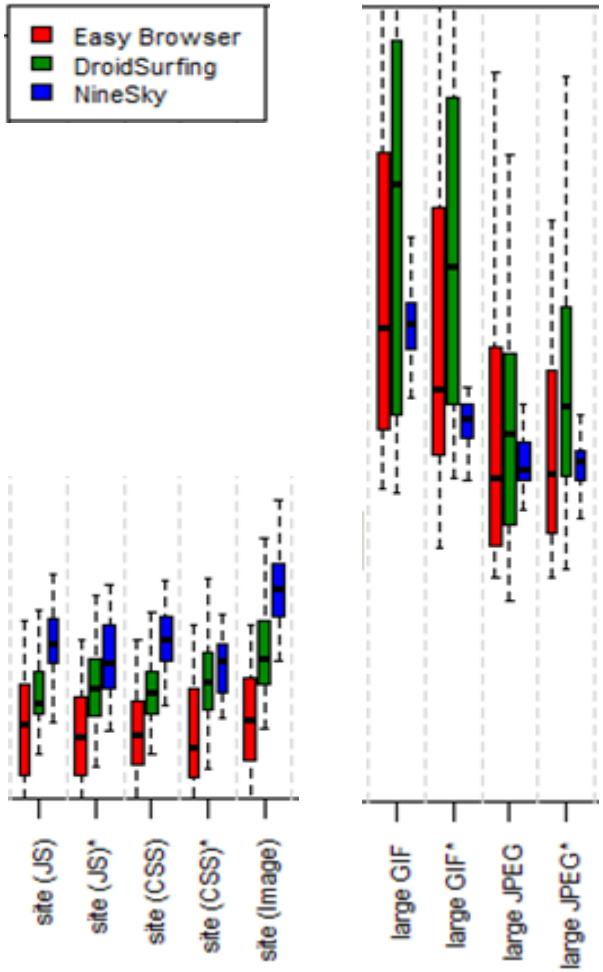


Median Web Browser Energy Consumption

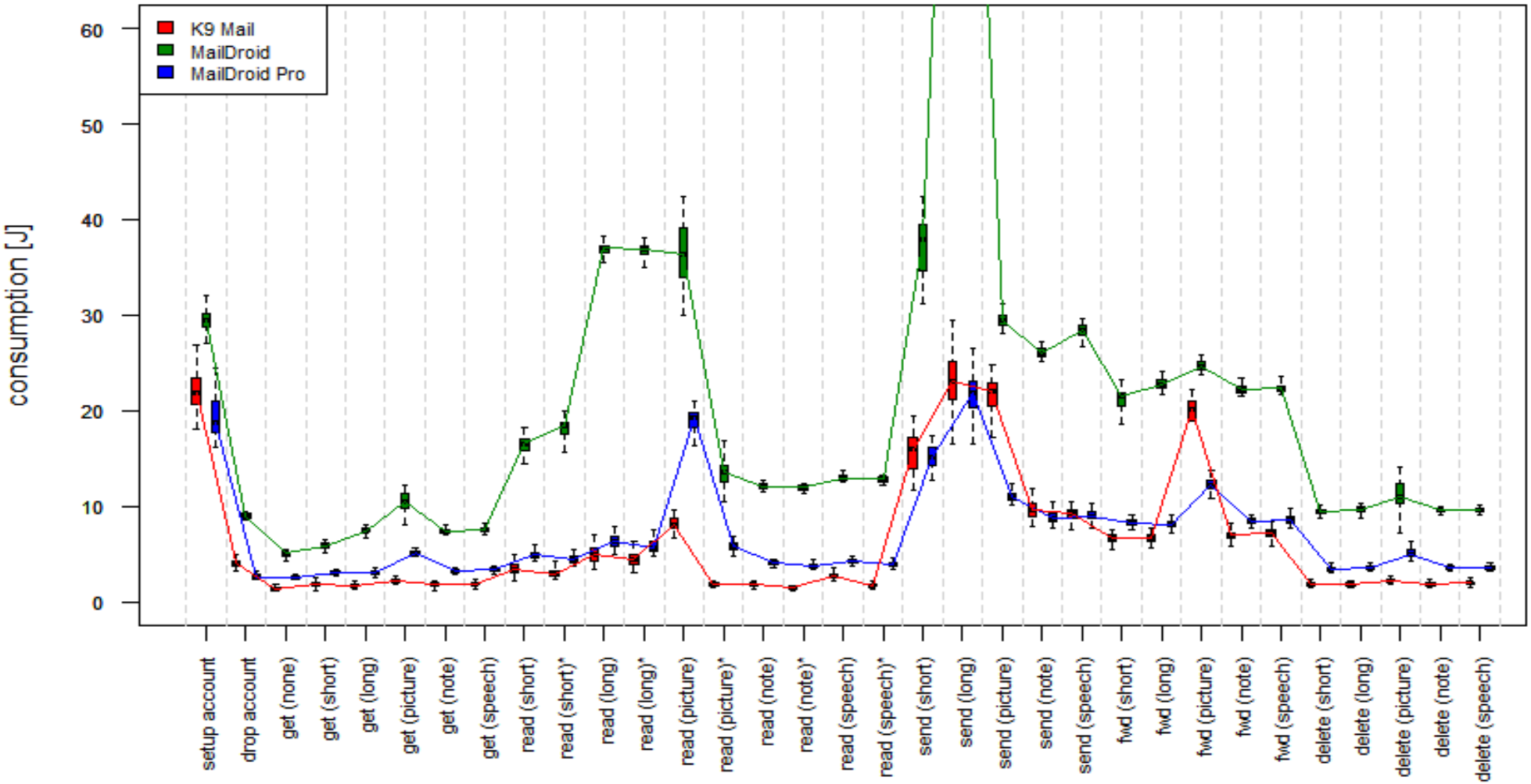


Interesting Issues with Web Browsers

- 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



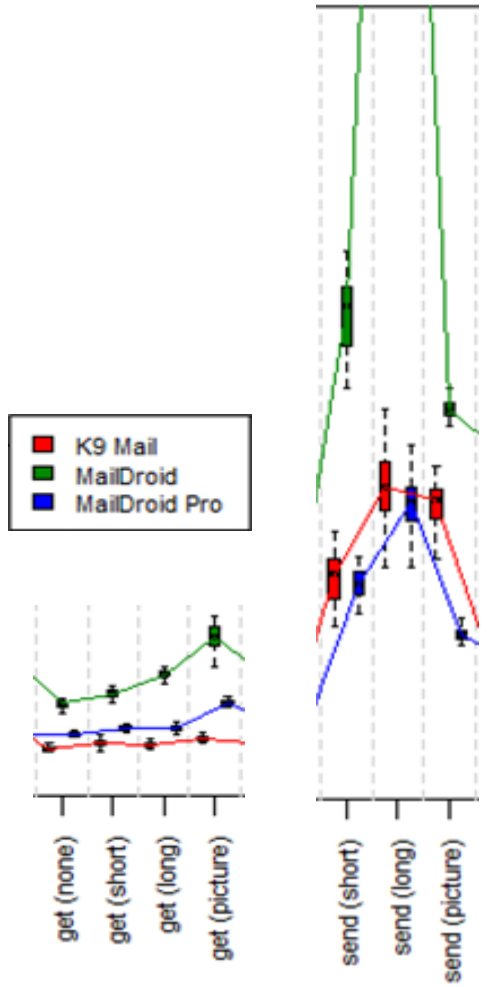
Median Mail Client Enery Consumption



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



12.1.3 Eclipse-Based Test Platforms

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

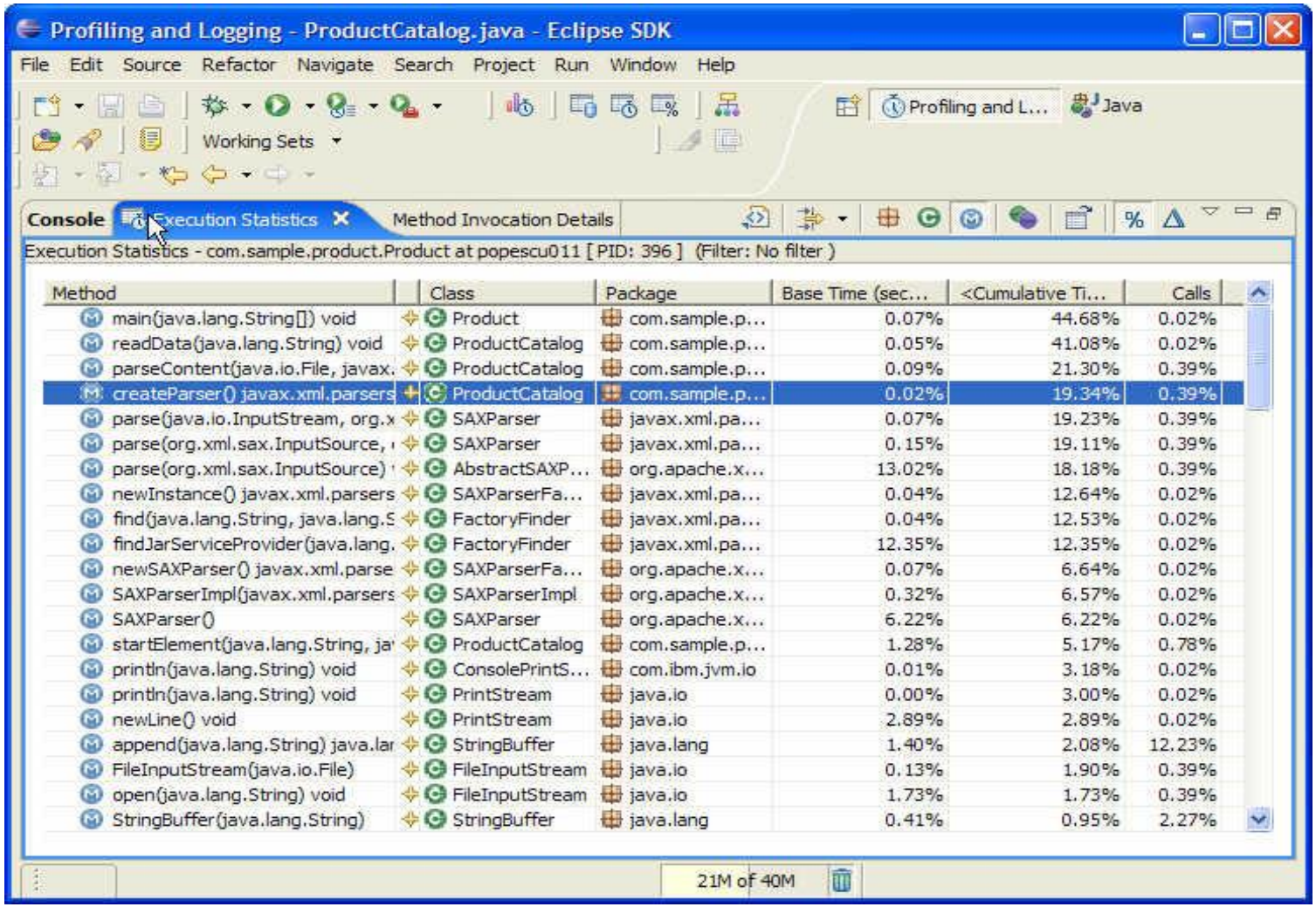
TPTP Testing Tools Project

- 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



12.2 Requirements-Oriented Testing

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

Test Environments

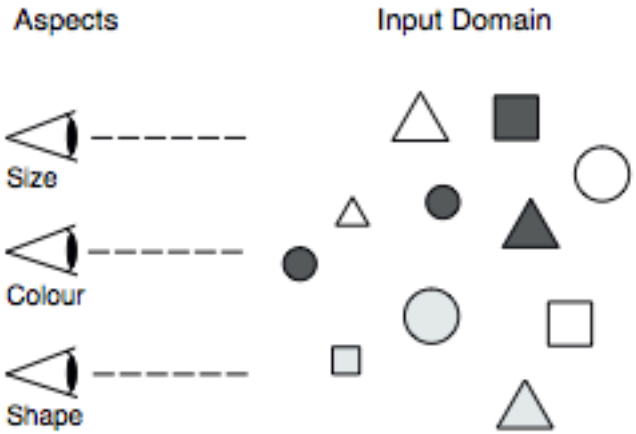
More: https://www.testtoolreview.de/de/

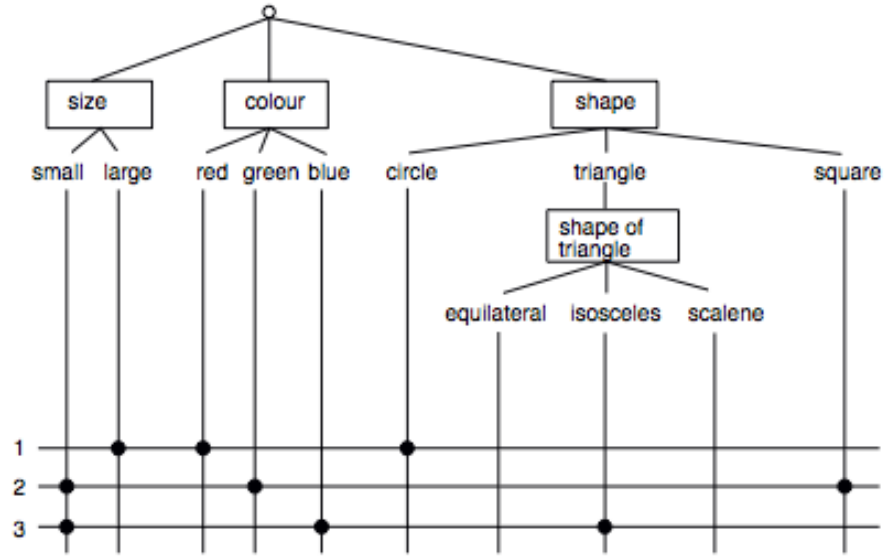
	Enterprise	URL
SilkTest	Segue Software	www.segue.com
TestBench	Imbus	www.imbus.de
Visual 2000	McCabe & Associates, USA	www.mccabe.com
Cantata++	IPL, Bath, UK	www.iplbath.com
ClickTracks	ClickTracks Analytics, Inc.CA	www.clicktracks.com
Tracetronic ECUTest	Tracetronic, Dresden. For motor tests	www.tracetronic.de

12.2.1 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*



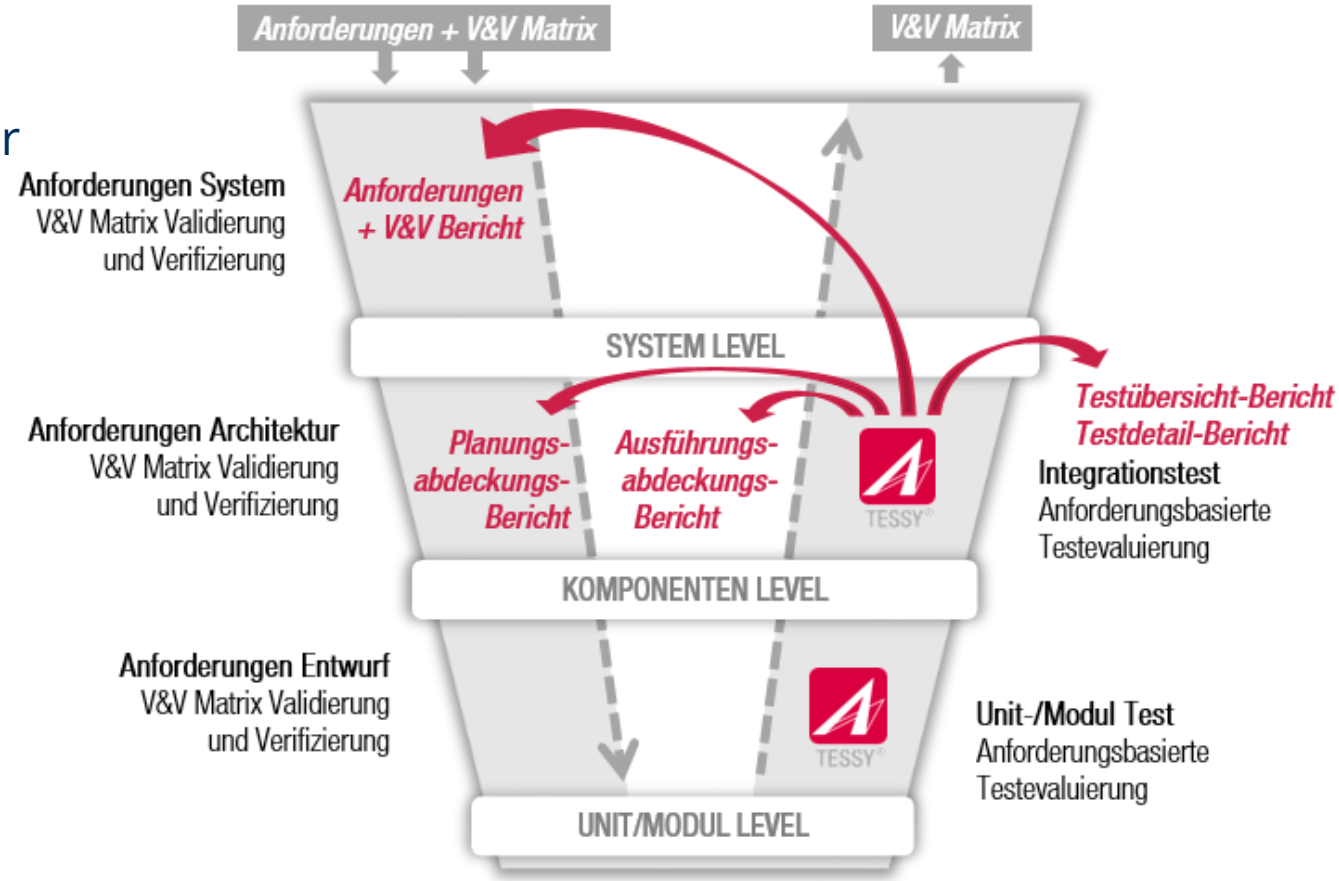


Advantages of Classification Tree Method

- 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

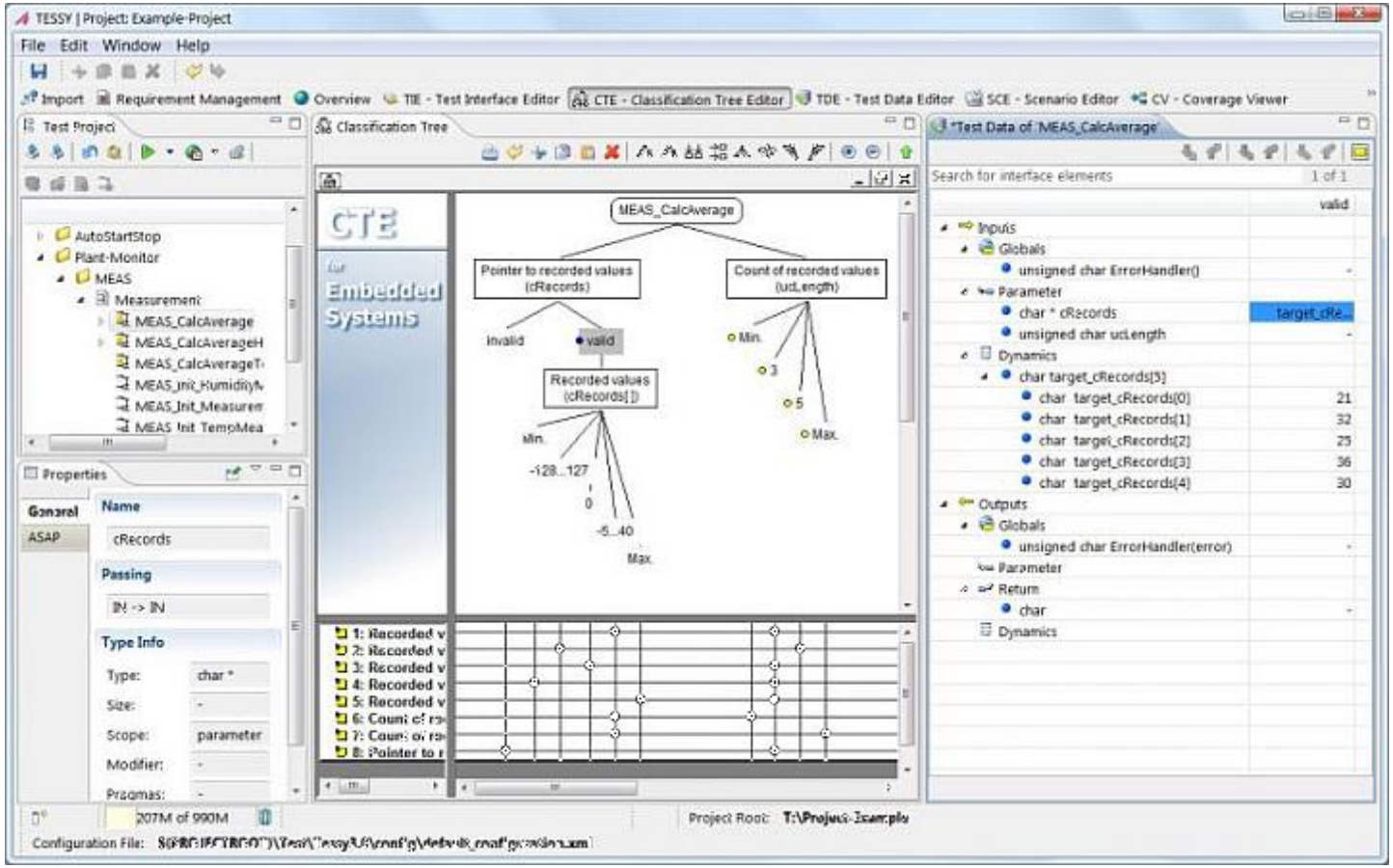
TESSY of HITEX

- 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



12.3.2 Imbus TestBench

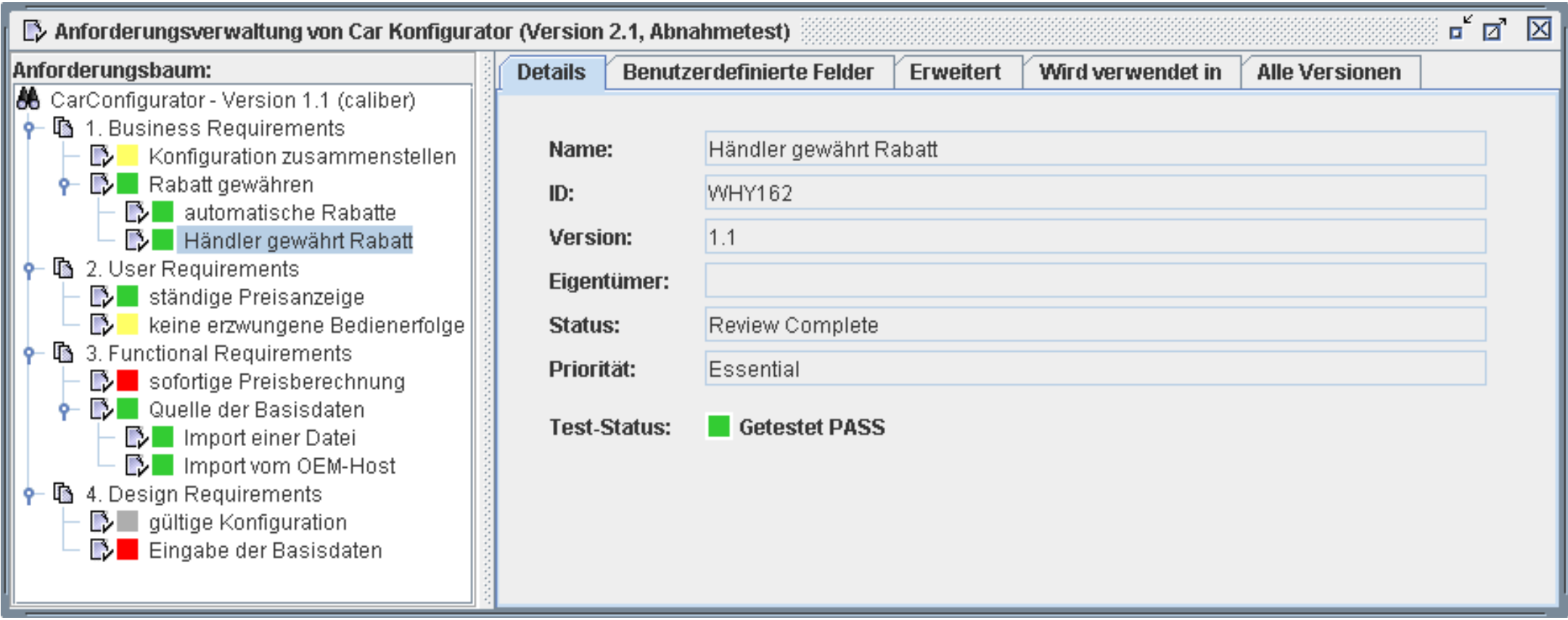
imbus TestBench

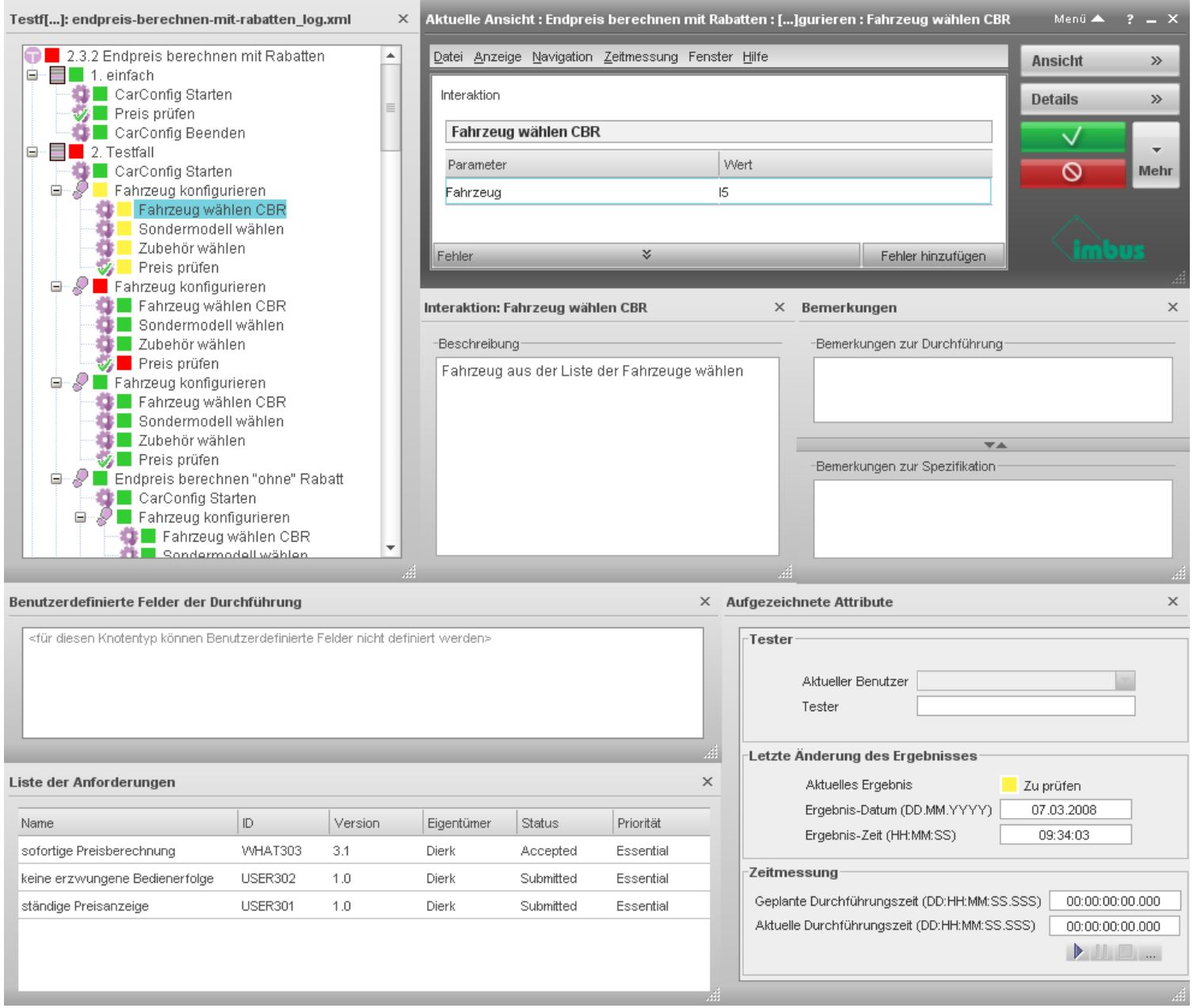
Imbus TestBench is a test environment supporting

- 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
Ansicht
Details
Interaktion
Fahrzeug wählen CBR
Parameter
Wert
Fahrzeug
I5
Mehr
Fehler
Fehler hinzufügen
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

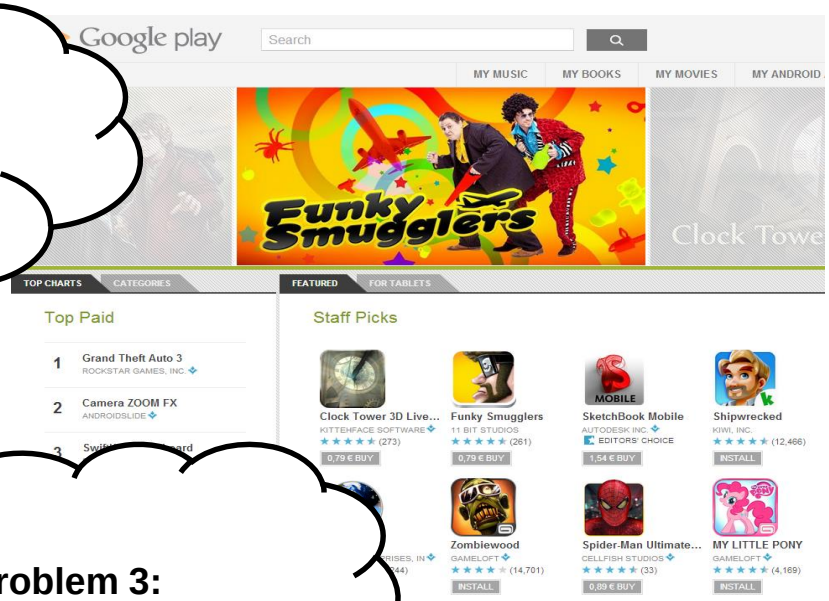
12.4 Model-Driven Test Environment MATE

Model-Driven Testing

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

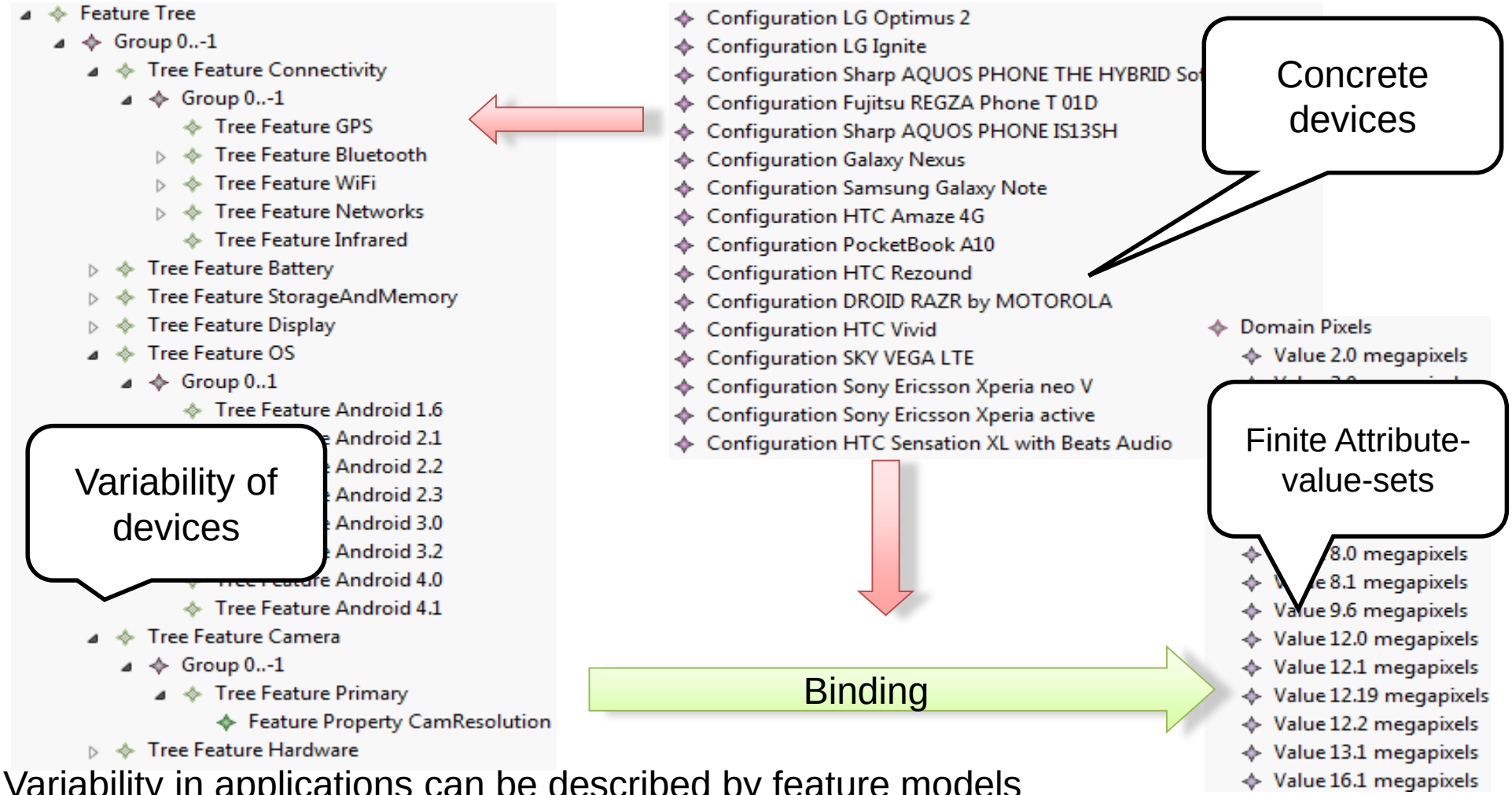
Problem 1:
Different platforms

Problem 2:
Apps differ on different devices



Problem 3:
Apps are context-adaptive

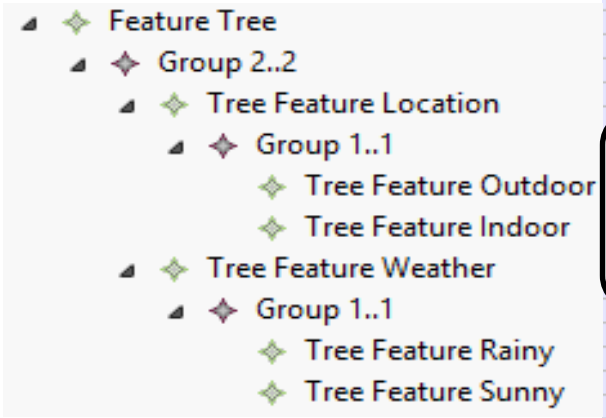
Plattform Management with Attributed Feature-Models (And-Or-Trees)

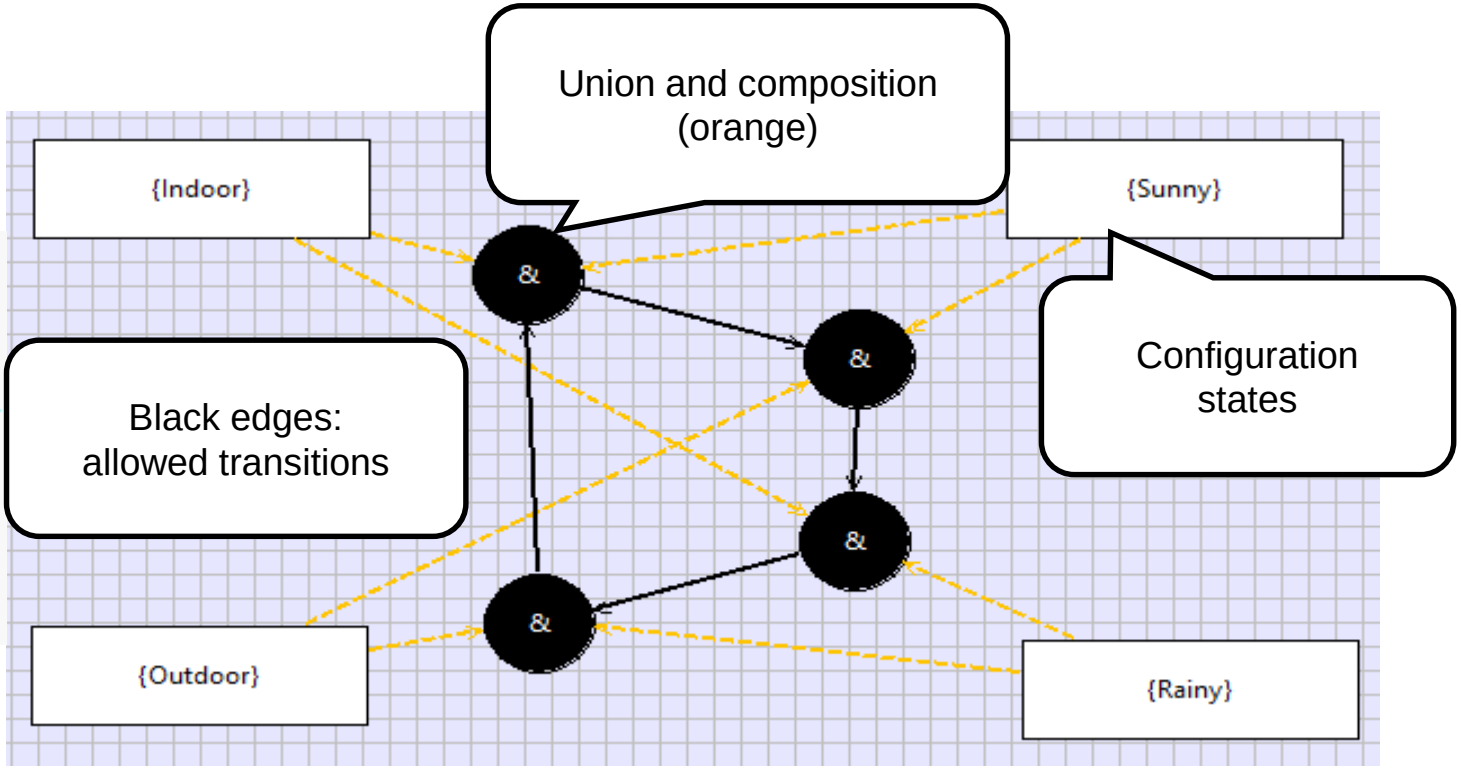


Variability in applications can be described by feature models

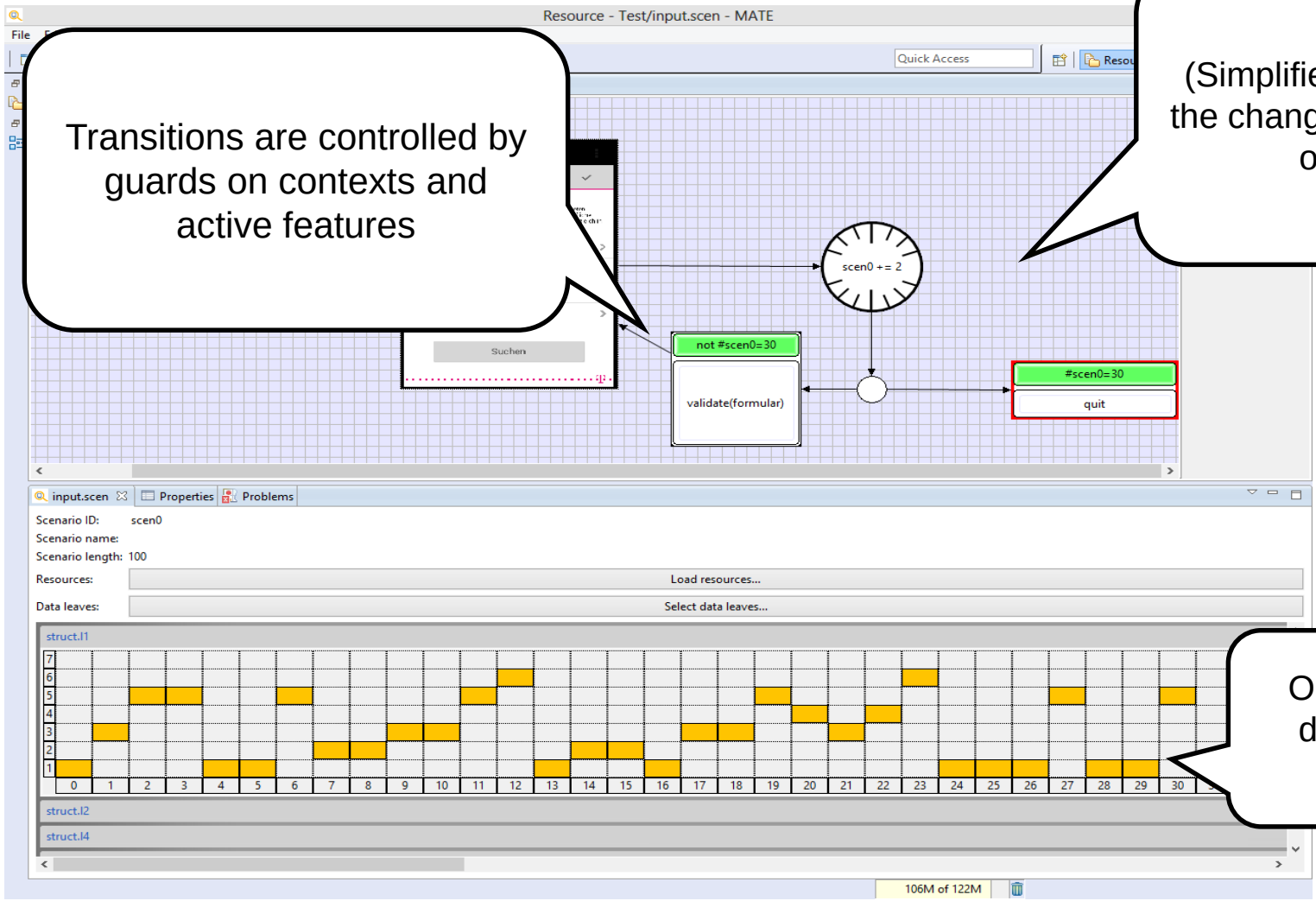
Dynamic Change of Features

- 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

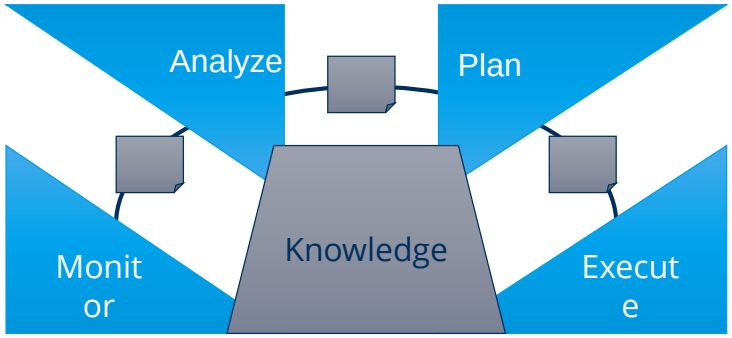
The MATE generator produces test cases by reachability analysis on the timed petrinet.

OCM alternative: Szenarios describe value changes of context data

Test of SAS must be Model-Based

Self-adaptive Systems (SAS) reconfigure themselfes

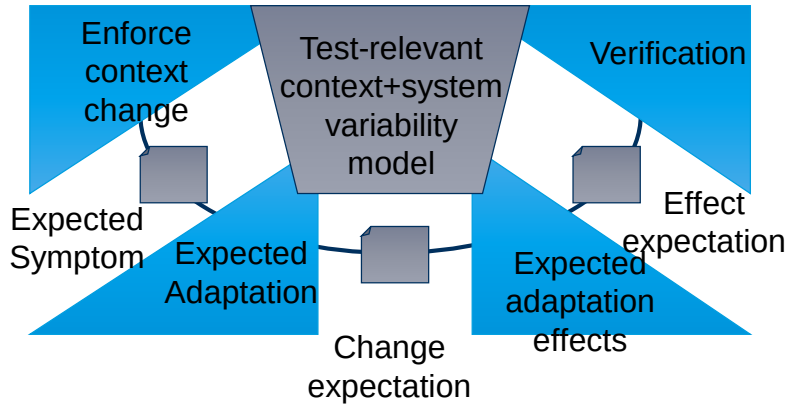
- at runtime
- according to requirements
- And dependent on the „context“, which may change over time.



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

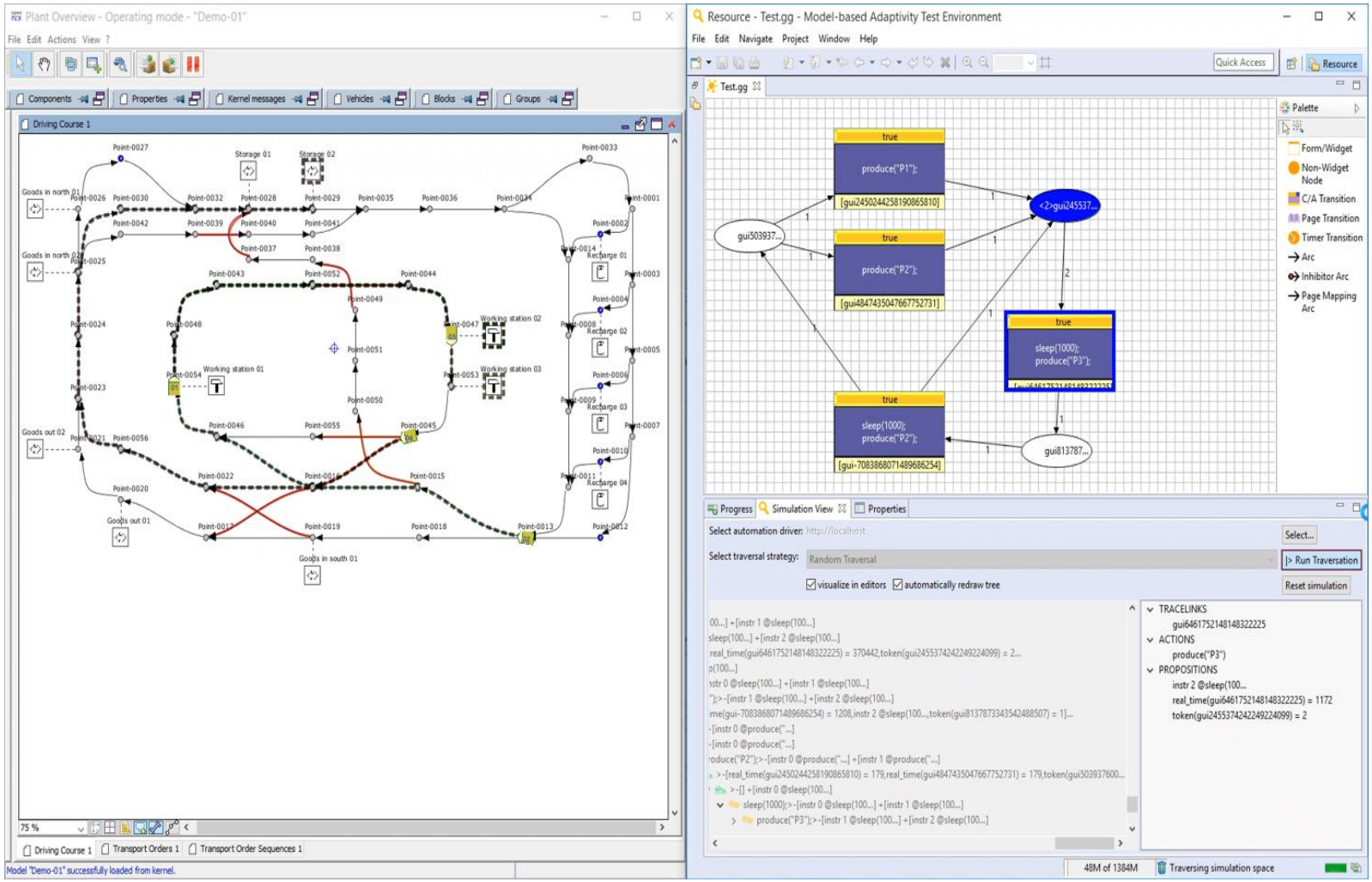
Solution A: Model-driven Testing (MDT)

- 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)

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



The End