TECHNISCHE UNIVERSITÄT DRESDEN

ResUbic

Qualitune

Fakultät Informatik Institut für Software- und Multimediatechnik - Lehrstuhl für Softwaretechnologie

JouleUnit

A Generic Framework for Software Energy Profiling and Testing

Claas Wilke, **Sebastian Götz**, Sebastian Richly

March 26th, 2013

Europa fördert Sachsen. ESF Europäischer Sozialfonds SACHSEN

Problem:

Software causes *too much*
energy consumption
for today's ICT applications!

Example 1: Smart Phones

“It started to drain the battery and having looooooooooooooong load times. This must be the worst update off all.”

(Glenn)

„Used 5% of my battery within five minutes.”

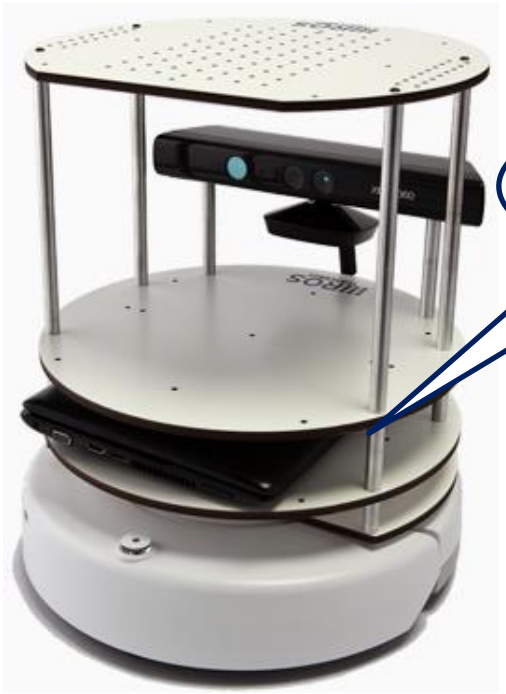
(Shawn)

„ I would like to logout during the night so it doesn't kill my battery.”

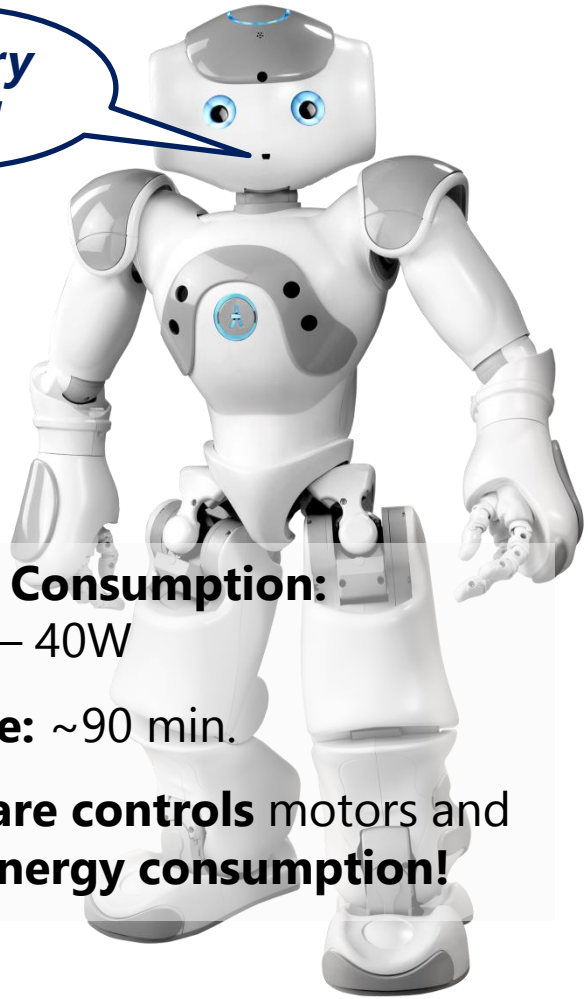
(Brittany)

http://www.hubba.com/wp-content/uploads/iStock_000017393041Small.jpg

Example 2: Roboter



Connection Lost



Battery Low!

- **Power Consumption:** ~20W – 40W

→ **Uptime:** ~90 min.

- **Software controls** motors and thus, **energy consumption!**

http://www.aldebaran-robotics.com/

Target:

Optimization *of software's* ***energy consumption***

Energy Profiling

- **Prerequesite of Optimization**
- **Many problem- and device-specific solutions exist**
  - Software- and hardware-based approaches [HSB12]
  - Focus on:
    - Desktop [LL06] [SEMN08],
    - Laptop [Fli01],
    - Server [KZL+10],
    - Smart phone [PHZ12] [VSF+12] [HSB12],
    - Robot applications [KW08].

- **But: no generic, reusable, easy-to-use solutions**

→ ***Target: a generic, reusable profiling framework***

REQUIREMENTS FOR ENERGY PROFILING

Profiling Requirements

/R1/ Workload capabilities

/R2/ Profiling capabilities

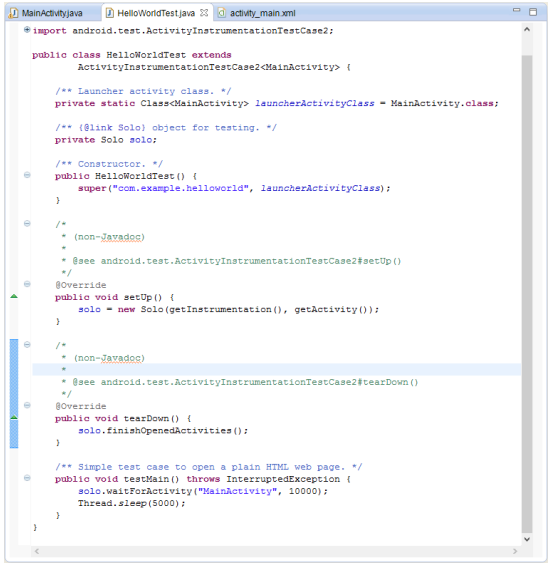
/R3/ Test run coordination

/R4/ Result evaluation and presentation

Requirements

/R1/ Workload capabilities

- **/R1-1/** Workload execution
- **/R1-2/** Setup and tear down functionality
- **/R1-3/** Event logging



Requirements

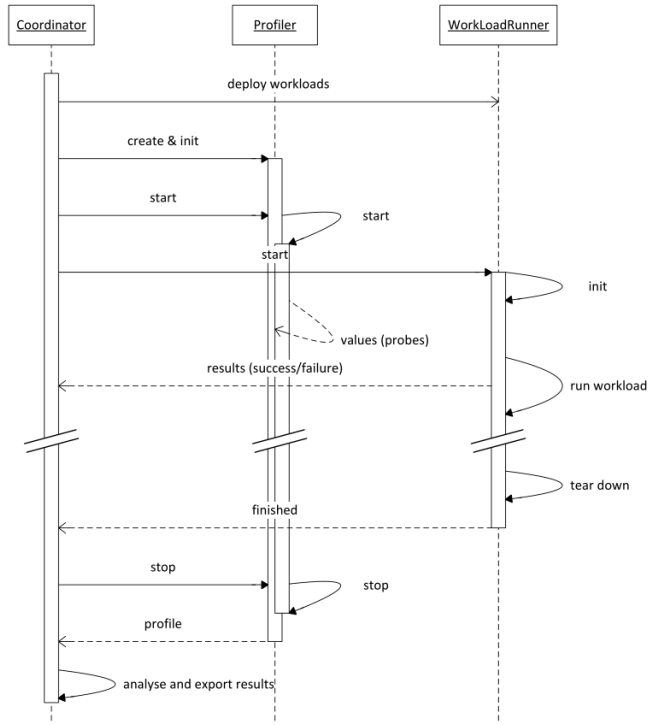
/R2/ Profiling capabilities

- **/R2-1/** Energy consumption profiling
- **/R2-2/** Automated profiler calibration
- **/R2-3/** Software- and hardware-based profiling
- **/R2-4/** On- and off-device profiling
- **/R2-5/** Eased integration of new devices
- **/R2-6/** Parallel profiling of multiple devices
- **/R2-7/** Profiling of further hardware information

Requirements

/R3/ Test run coordination

- **/R3-1/** Coordinated profiler and workload execution
- **/R3-2/** Multiple runs
- **/R3-3/** Synchronization of workload and profiling events
- **/R3-4/** Synchronization of system clocks

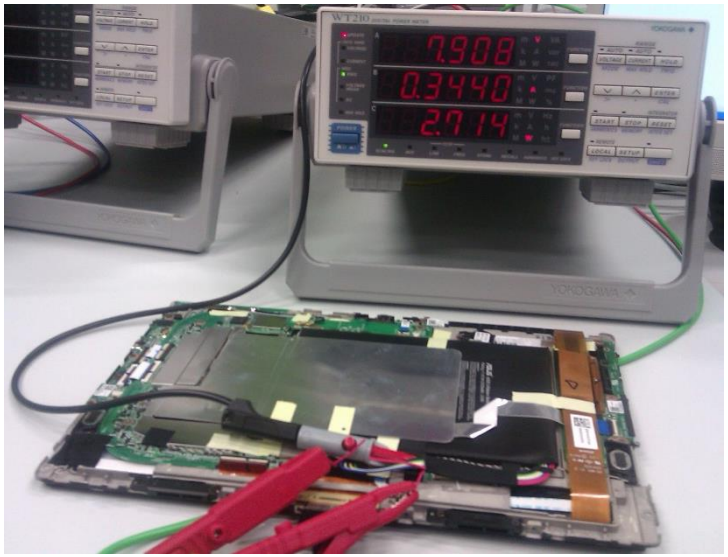


Requirements

/R4/ Result evaluation and presentation

- **/R4-1/** Result evaluation
- **/R4-2/** Result preservation
- **/R4-3/** Result presentation

JOULE UNIT



JouleUnit
Top-level architecture

JouleUnit Workbench
Concepts to trigger profiling, and to preserve, export and present profiling results.

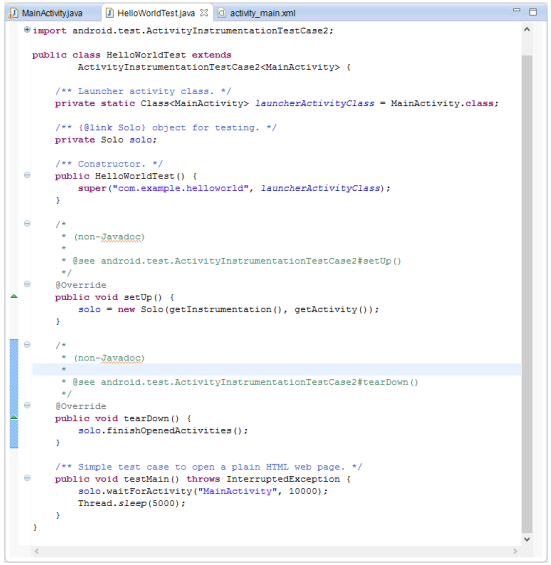
JouleUnit Coordinator
Concepts to deploy and execute workloads, coordinate profiling and evaluate results.

Workload Runner
Concepts to define use cases and data, as well as for their execution.

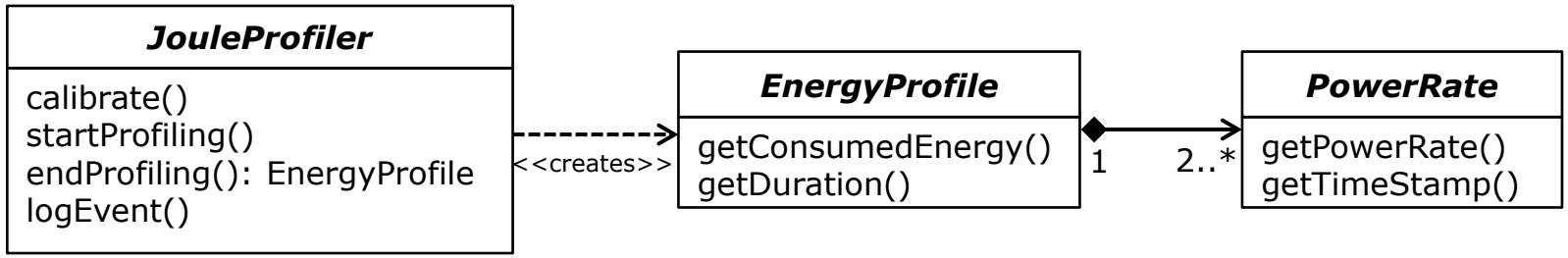
Energy Profiler
Concepts to pick power rate probes and compute energy consumption.

The workload runner

- **Reuses the JUnit framework**
  - Workload execution (/R1-1/)
  - Setup and tear down (/R1-2/)
- **Event logging** must be provided **device/OS-specific** (/R1-3/)
  - E.g., by reusing Android Log Cat



The energy profiler



- **Profiling and automated calibration** (/R2-1/, /R2-2/)
- **On-/Off-device profiling based on deployment** (/R2-3/, /R2-4/)
- **Abstract reusable classes**
  → Eased integration of new devices, plattforms (/R2-5/)
- **Composite profilers** supported (/R2-6/)
- **Further hardware profiling possible** (/R2-7/)

The coordination layer

- **Encapsulates workflow for coordinated** workload and profiler **execution** (/R3-1/)
  - Abstract implementation provides workflow sceleton
  - Support of multiple runs (/R3-2/)

JouleUnitCoordinator
+ runTestRun() # computeTimestampOffsets() # deployTestCases() # startEnergyProfiling() # startHardwareProfiling() # runTestCases() # stopHardwareProfiling() # stopEnergyProfiling() # undeployTestCases() # propagateResults()

- **Synchronization of workload and profiling events** based on timestamps (/R3-3/)
- **Automated** NTP-based **system clock synchronization** (/R3-4/)

The JouleUnit workbench

- **Result evaluation** (/R4-1/)
  - Computation of average results and standard deviation
- **Result preservation** (/R4-2/)
  - CSV and SQL export
- **Result presentation** (/R4-3/)
  - Eclipse-integrated runners and views
  - Based on abstract concepts
  - → Reusable for different devices and OSs

Joule Unit - Eclipse Platform

File Edit Refactor Source Navigate Search Project Run Window Help

Quick Access

Project Explorer

- asl-demo
- Chrome
- com.maildroid.test
- com.ninesky.browser.test
- com.opera.browser.test
- com.ovmobile.droidsurfing.test
- droidSurfing
- easy.browser.classic.test
- EasyBrowser
- FirefoxTesting
- k9mail
- k9mail.test
- Maxthon
- Ninesky
- > OperaTest 2378 [svn+ssh://svn-st.inf.tu-dresc
- org.mozilla.firefox.test
- org.qualitune.jouleunit.android.hwservice
- > org.qualitune.jouleunit.android.tests.browser
- org.qualitune.jouleunit.android.tests.mail 2235
- > org.qualitune.jouleunit.tests.k9mail 1911 [svn
- org.qualitune.jouleunit.tests.maildroid 2290 [svi
- org.qualitune.jouleunit.tests.maildroidpro 2373
- simple.sort.example
- video.example.simple 2496 [svn+ssh://svn-st.in

JUnit

Finished after 30,844 seconds

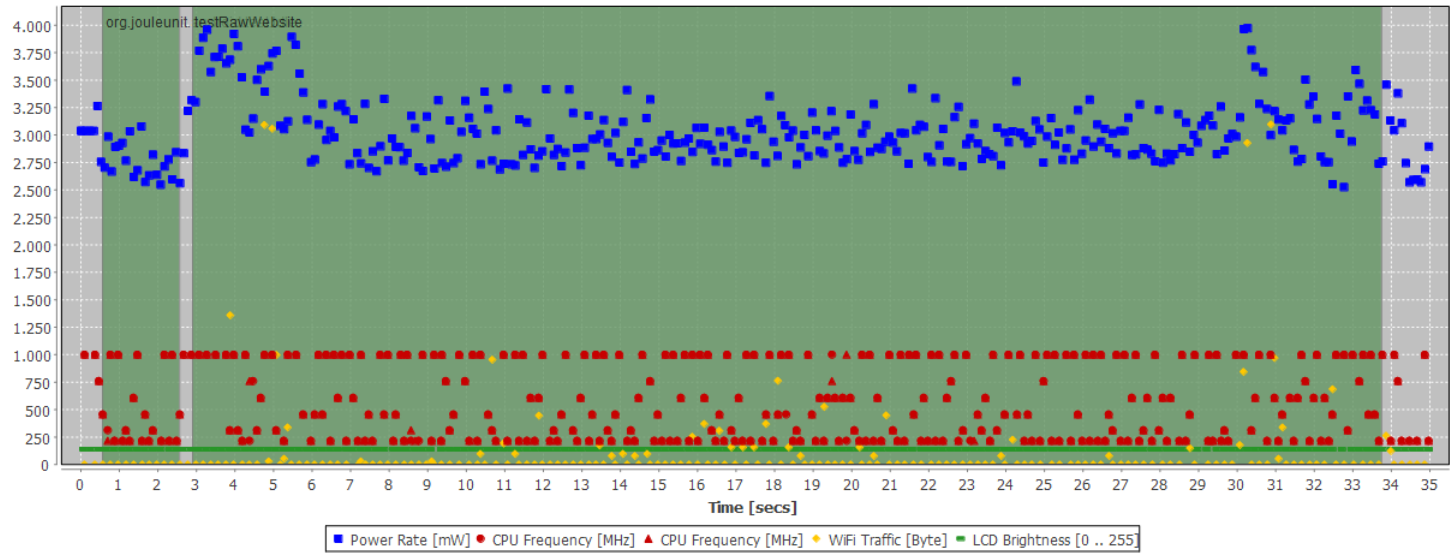
Runs: 1/1 Errors: 0 Failures: 0

easy.browser.classic.test.EasyBrowserTests [Run

Failure Trace

Test Run Details

Show values from sec. 0.0 to sec. 35.0 Update Chart



Test Run Results

Individual Test Case Results | Avg. Test Case Results

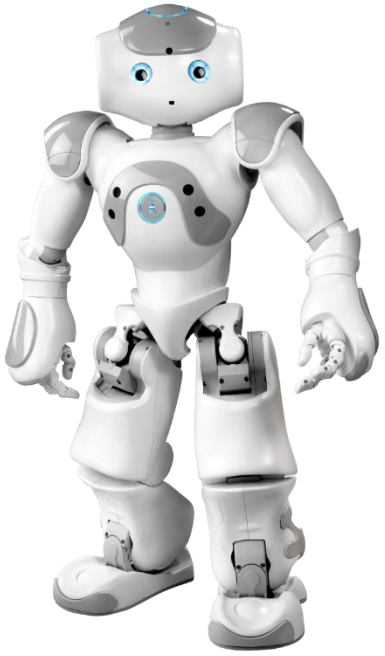
Test Case	Start [ms]	Stop [ms]	Duration [ms]	Avg. Power Rate [mW]	Energy Consumption [mJ]
org.jouleunit.android.test.Idle	1350993670179	1350993672178	1999,00	2764,28	5525,79
testRawWebsite(easy.browser.classic.test.EasyBro...	1350993672513	1350993703363	30850,00	3046,41	93981,65

Console

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.
```

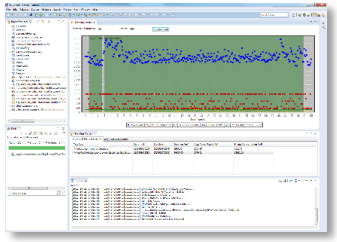
CASE STUDIES



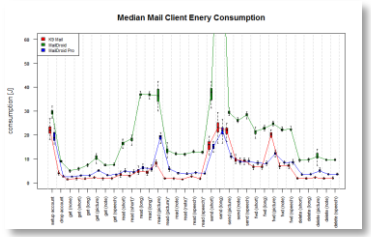
JouleUnit for Android

- **Workload runner**
  - Reuse of the JUnit runner for Android
- **Energy profiler**
  - Software-based profiling (ACPI infos from /proc)
  - Hardware-based profiling (Yokogawa WT210 power meter)
- **Coordination Layer**
  - Android-specific implementation
  - Based on the Android Debug Bridge (ADB) and Log Cat

Comparing Android apps' energy consumption



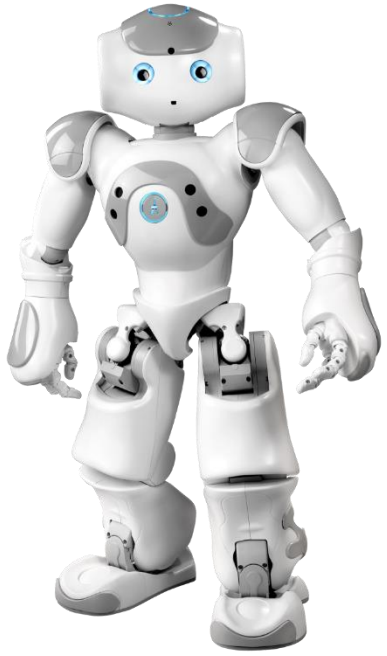
- Definition of **benchmarks** as a set of **test cases**
  - **Web browsing**
  - **Emailing**
- **Instantiation** of the benchmarks for **existing applications** [WRP+12] [WRG+13]
  - EasyBrowser, DroidSurfing, NineSky
  - K9 Mail, MailDroid, MailDroid Pro
- Profiling of **each use case 50 times** per application



→ ***The first approach comparing apps from usage domains!***

JouleUnit for NAOs

- **Workload runner**
  - Workload execution via web service [GLR+12]
- **Energy profiler**
  - Power rates from NAO's battery sensor (20 Hz)
  - Profiling on device, retrieval via web service
- **Coordination Layer**
  - Simple Java implementation triggering JUnit and profiler
- **Overall implementation time:** 3-4 hours



Lessons learnt

- **Eased implementation for individual devices**
  - Sensible layering of framework
  - Reuse of abstract implementations
- **Reuseable coordinator implementation**
  - Saves a lot of time
  - No worries on time synchronization

Summary

- **Energy optimization of software is important**
- **Many profiling approaches, but no generic solutions**
- **Requirements analysis**
- **JouleUnit**
  - Implementations for Android and NAOs
  - http://www.jouleunit.org/
- **Further questions?**
  - claas.wilke@tu-dresden.de

References

[Fli01] J. Flinn. *Extending Mobile Computer Battery Life through Energy-Aware Adaptation*. PhD thesis, Carnegie Mellon University, 2001.

[GLR+12] S. Götz, M. Leuthäuser, J. Reimann, J. Schroeter, C. Wende, C. Wilke, and U. Aßmann. NaoText: *A Role-based Language for Collaborative Robot Applications*. In Leveraging Applications of Formal Methods, Verification, and Validation, vol. 336 of CCIS, Springer (2012), pp. 1–15.

[HSB12] Höpfner, H., Schirmer, M., Bunse, C.: *On Measuring Smartphones' Software Energy Requirements*. In: ICSOFT2012, SciTePress (2012), pp. 165–171.

[KZL+10] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. Bhattacharya. *Virtual machine power metering and provisioning*. In 1st ACM symposium on Cloud computing, ACM (2010), pp. 39–50.

[KW08] J. A. Kulk and J. S. Welsh. *A low power walk for the nao robot*. In ACRA 2008, 2008.

References

[LL06] S. Lafond and J. Lilius. *An Energy Consumption Model for an Embedded Java Virtual Machine*. In ARCS 2006, vol. 3894 of LNCS, Springer (2006), pp 311–325.

[PHZ12] A. Pathak, Y. C. Hu, and M. Zhang. *Where is the energy spent inside my app?: fine grained energy accounting on smartphones with Eprof.* In 7th ACM European Conference on Computer Systems, ACM (2012), pp. 29–42.

[SEMN08] C. Seo, G. Edwards, S. Malek, and N. Medvidovic. *A Framework for Estimating the Impact of a Distributed Software System's Architectural Style on its Energy Consumption.* In WICSA 2008, IEEE (2008), pp. 277–280.

[VSF+12] N. Vallina-Rodriguez, J. Shah, A. Finamore, Y. Grunenberger, H. Haddadi, and J. Crowcroft. *Breaking for Commercials: Characterizing Mobile Advertising.* In IMC 2012. ACM (2012).

References

[WRP+12] Wilke, C., Richly, S., Püschel, G., Piechnick, C., Götz, S., Aßmannn, U. *Energy Labels for Mobile Applications.* In: EEbS 2012, GI (2012), pp. 93–102.

[WRG+13] Claas Wilke, Sebastian Richly, Sebastian Götz, Christian Piechnick, Georg Püschel und Uwe Aßmann. *Comparing Mobile Applications' Power Consumption.* In: SEGC Track at SAC2013, ACM (2013).

TECHNISCHE UNIVERSITÄT DRESDEN

ResUbic

Qualitune

Fakultät Informatik Institut für Software- und Multimediatechnik - Lehrstuhl für Softwaretechnologie

JouleUnit

A Generic Framework for Software Energy Profiling and Testing

Claas Wilke, **Sebastian Götz**, Sebastian Richly

March 26th, 2013

Europa fördert Sachsen. ESF Europäischer Sozialfonds SACHSEN