2. Workshop: Energy Aware Software-Engineering and Development (EASED@BUIS)

Entwicklung und Klassifikation energiebewusster und energieeffizienter Software

Christian Bunse, Stefan Naumann und Andreas Winter

Zusammenfassung Der Energieverbrauch von Computer-Systemen unterliegt einem kontinuierlichem Wachstum. Bereits 2006 sagte das statistische Amt der Europäischen Union voraus, dass im Jahr 2020 bis zu 20% des weltweiten Energiebedarf auf Informationstechnik zurückzuführen sein wird. Aktuelle Forschungsarbeiten zeigen, dass neben der eingesetzten Hardware insbesondere auch die ausgeführte Software ein wichtiger Einflussfaktor auf den Energiebedarf der Informationstechnik ist. Dieses Papier beschäftigt sich mit dem Themenkomplex *Energiebewusste und energieeffiziente Software* und zeigt dessen Bandbreite auf. Das Papier zielt hierbei auf einen allgemeinen Überblick zur Energieeffizienz in der Softwaretechnik und versucht eine erste Begriffseingrenzung. Einen besonderen Schwerpunkt nehmen hierbei die Themenkomplexe ein, welche im Rahmen des 2. Internationalen Workshops „Energy Aware Software-Engineering and Development (EASED@BUIS)“, der im Rahmen der BUIS-Tage 2013 durchgeführt wird, diskutiert werden. Herausforderungen bei der Betrachtung der Energieeffizienz aus Sicht der Softwaretechnik sind insbesondere in der Messung und Abschätzung des durch Software verursachten Energiebedarfs zu sehen. Darüber hinaus interessieren Methoden und Verfahren zur Erstellung energieeffizienter Software, zur Optimierung des Software-induzierten Energieaufwands sowie zur energetischen Zertifizierung und Klassifizierung von Software-Applikationen.

Christian Bunse
Fachhochschule Stralsund, Stralsund e-mail: Christian.Bunse@fh-stralsund.de

Stefan Naumann
Hochschule Trier, Umwelt-Campus Birkenfeld e-mail: s.naumann@umwelt-campus.de

Andreas Winter
Carl von Ossietzky Universität Oldenburg, Softwaretechnik e-mail: winter@se.uni-oldenburg.de

1 Einleitung

Mobile und eingebettete Geräte sind ein wichtiger und dynamischer Trend der Informations- und Kommunikationstechnologie (IKT). Die fortschreitende Verkleinerung von Geräten und deren zunehmend flexiblere Nutzung und Programmierung erleichtert ihre inzwischen ubiquitäre Präsenz. Ein limitierender Faktor bei der Verbreitung von IKT ist deren begrenzte Verfügbarkeit von Ressourcen, insbesondere im Bereich der Energieversorgung, bei gleichzeitig hohen bzw. steigenden Leistungsanforderungen. Somit besteht ein starker Zusammenhang zwischen erreichbarer Gerätelaufzeit und Effektivität der bereitgestellten Dienste. Geeignete effiziente Energie-Spartechniken, die direkt auf die Optimierung des Energieverbrauchs der Softwarekomponenten mobiler IKT zielen, können zur Erhöhung von Gerätelaufzeiten beitragen. Aber auch über mobile und eingebettete Systeme hinaus besteht aus ökologischer und ökonomischer Perspektive der Bedarf, energiebewusste Software zu entwickeln, um den ökologischen Fußabdruck der IKT zu verringern.

Software dient dazu, bestimmte Aufgaben zu erfüllen. Dabei unterscheiden sich die dazu verwendeten Hardware-Komponenten teilweise gravierend hinsichtlich ihres Energiebedarfs. Üblicherweise optimieren IKT-Systeme ihren Energiebedarf durch eine Anpassung der Hardware-Nutzung, zum Beispiel durch Reduktion der CPU-Taktrate oder durch geeignete Nutzung von Virtualisierungen.

Neben solchen Ansätzen ist es jedoch auch möglich, den Energiebedarf mittels Adaption der Software selbst zu optimieren. Ansätze hierzu umfassen die Nutzung sog. Ressourcen-Substitutionsstrategien zur Einsparung mittels Austausch/Substitution von Service-Providern, die Optimierung der Software selbst oder die Ausnutzung von Informationen über das (erwartete) Benutzerverhalten. Allerdings sind die exakten Zusammenhänge von Energie, Performanz, Kosten und anderen Metriken noch weitgehend unbekannt, so dass Erfolge und Verbesserungsoptionen solcher Maßnahmen nur schwer nachweisbar sind. Hier entsteht ein neues Forschungsfeld hinsichtlich der Messung und Klassifizierung von Software-Energiebedarfen sowie der Entwicklung von Verfahren zur Verbesserung der Energieeffizienz. Aktuelle Themenkomplexe hierbei sind insbesondere:

- Entwicklung von Verfahren zur Messung und Abschätzung von Energie-Bedarfen von bzw. für Softwarekomponenten
- Ableitung von Energiemodellen zur Abbildung des Energiebedarfs von Software-Systemen und deren Spezifikation
- Entwicklung von standardisierten Energie-Benchmarks zur wiederhol- und vergleichbaren Messung des Energiebedarfs von Software-Systemen
- Bereitstellung von Methoden und Verfahren zur Entwicklung energiebewusster und -effizienter Software-Systeme
- Entwicklung von Techniken zur Erkennung von „Energieverschwendung“ durch Software-Komponenten
- Definition von Klassifikations- oder Zertifizierungsschemata für energiebewusste Software
- Entwicklung von Strategien zur Optimierung des Software-Energiebedarfs

- Vorstellung von Mechanismen der energieeffizienten Kommunikation
- Entwicklung energiebewusster, -effizienter Software und Green Design Patterns

Im Folgenden wird zunächst der Begriff *energiebewusste Software* eingegrenzt (Abschnitt 2). In Abschnitt 3 wird detaillierter auf den Bereich des Messens der Energiebedarfe von Softwaresystemen eingegangen. Hierauf aufbauend werden die Integration von Energieaspekten in Software-Entwicklungstechniken, -methoden und -werkzeuge (Abschnitt 4) sowie die Nutzung dieser Ansätze zur Vorhersage und Optimierung (vgl. Abschnitt 5) betrachtet. Zertifizierungs- und Klassifikationsschemata werden in Abschnitt 6 kurz diskutiert, während Abschnitt 7 die Ergebnisse zusammenfasst und diese in den Kontext des *2. Workshops „Energy Aware Software-Engineering and Development (EASED@BUIS)*“ einbettet.

2 Energiebewusste Software

Für das Forschungsfeld der energiebewussten und -effizienten Software (energy-aware and energy efficient software) gibt es bislang keine einheitlich akzeptierte Definition. Aus der Perspektive einer nachhaltigen Informationstechnik lassen sich zwei Gebiete unterscheiden: *Green IT* bezeichnet die „*Entwicklung und Anwendung von Verfahren zur Bewertung, Analyse und Steigerung der Energieeffizienz von IT-Systemen*“[1], beispielsweise durch sparsame Hardware, Virtualisierungen und Softwareoptimierungen. Unter *Green by IT* oder *IT2Green* werden Lösungen der Informatik verstanden, welche in anderen Bereichen zu höherer Energieeffizienz und mehr Nachhaltigkeit führen. Hierunter fallen bspw. Betriebliche Umweltinformationssysteme, Algorithmen zur Minimierung von Transportstrecken im Logistikbereich etc.

Energiebewusste Software ist dem Bereich Green IT zuzuordnen. Hierunter wird Software verstanden, die neben klassischen Benchmarks wie Laufzeit, CPU- und Speicherbedarf auch explizit den durch die Software induzierten Energieverbrauch in den Fokus nimmt. Dabei ist zu beachten, dass der Energieverbrauch von Software unmittelbar von den Nutzungsszenarien der Software und von der Interaktion mit umgebender Soft- und Hardware (bspw. Betriebssystem und Software-gesteuerte Hardwarekomponenten) abhängig ist.

Energiebewusste Software bezeichnet im Folgenden Software,

- deren Energieverbrauch anhand nachvollziehbarer und wiederholbarer Kriterien gemessen und nachvollziehbar dargestellt werden kann,
- die bereits während der Software-Entwicklung und in der Software-Evolution vor dem Hintergrund der Energieeffizienz erstellt und weiterentwickelt wird, und
- deren Energiebedarf, soweit möglich, fortlaufend während des Betriebs und der Evolution verringert wird.

[1] http://www.informatik.uni-oldenburg.de/48646.html

3 Messung von Software-Energiebedarfen

Forschungsergebnisse (Bunse und Höpfner, 2010; Bunse et al, 2011; Höpfner und Bunse, 2010b) zeigen eine eindeutige Korrelation zwischen der auf einem System ausgeführten Software und dem Energiebedarf dieses Systems. Im Umkehrschluss ist es somit möglich, durch Modifikation der Software eine Reduktion des Energiebedarfs zu erreichen. Grundlage jeglicher Arbeiten ist dabei die präzise und verlässliche Messung des Energiebedarfs von Software. Dies erfordert ein Instrumentarium bzw. Messinstrumente für die Erhebung und Evaluierung der Energiebedarfe und das Erreichen von Optimierungszielen. Messungen müssen dabei zeitnah erfolgen, eindeutig einer Software bzw. deren Komponenten zuzuordnen sein und, wenn auch nicht in absoluten Zahlen, plattformübergreifend und für vergleichbare Software-Produktfamilien (z. B. für unterschiedliche Webbrowser oder Textverarbeitungssysteme) wiederholbar sein (Höpfner et al, 2012).

Die systematische Erfassung bzw. Messung von Energiebedarfen erfolgt indirekt über die Erfassung von Spannungswerten (Höpfner und Bunse, 2010a). Mit Hilfe der Ohmschen und Kirchhoffschen Gesetzen kann dann der Energiebedarf (J) durch Berechnung der Fläche unter Integral der Messdaten ermittelt werden. Spannungswerte können dabei auf zwei verschiedenen Wegen gewonnen werden (vgl. auch (Josefiok, 2012)):

1. Anwendung von Messverfahren aus der Elektrotechnik, bei der Spannungswerte direkt an der entsprechenden Hardware-Komponente abgenommen werden (offline-Messung). Vorteile eines derartigen Vorgehen sind hohe Messfrequenzen (im usec Bereich) sowie die Möglichkeit, die Bedarfe einzelner Hardware-Komponenten direkt zu ermitteln. Nachteile sind hohe Kosten, die isolierte Betrachtung einzelner Geräte und die Schwierigkeit Bedarfe Software-Artefakten zuzuordnen. Ebenso erfordern diese Messungen ggf. ein Öffnen der zu vermessenden Hardware (z B. bei durchgängig verschweißten Mobiltelefonen).
2. Alternativ bieten moderne Betriebssysteme (z. B. Android) die Möglichkeit mittels Software (online-Messung) Werte wie beispielsweise den Ladezustand der Batterie oder die Höhe des aktuell entnommenen Stroms zu ermitteln. Dies erlaubt die Entwicklung von kostengünstigen Messverfahren, die über eine Vielzahl von Systemen hinweg genutzt werden können. Allerdings sind die erfassten Werte häufig Näherungswerte, die oft nur im Minutentakt ermittelt werden. Teilweise sind Rückschlüsse auf einzelne Verbraucher wie etwa Display oder Speicher nicht zu ziehen. Diese „Software-gestützten Messverfahren“ basieren oft auf Energiemodellen, die — vom Hardware-Hersteller geliefert — Verbrauchsdaten einzelner Hardware-Komponenten abbilden. Aus diesen Daten und den häufig feingranular messbaren Nutzungsdauer der Komponenten lassen sich bei passenden Energiemodellen häufig hinreichende Abschätzungen des Energiebedarfs ableiten.

Auf ausgewählten Plattformen hat sich gezeigt, dass die Ergebnisse beider Messansätze durchaus vergleichbar sind (Höpfner et al, 2012). Weiterhin stellt sich im Rahmen des Themenkomplexes Energiebedarf von Software die Frage nach

der notwendigen Granularität und Präzision von Messdaten. Werden Daten im Mikrosekunden-Takt und hoher Genauigkeit tatsächlich benötigt oder reicht die Ermittlung von Energieklassen bzw. Größenordnungen? Die Antwort hierauf wird bedingt durch den späteren Einsatzzweck sowie den Abstraktionsgrad der Betrachtung (Statement, Komponente, System, etc.).

4 Entwicklung energiebewusster Software

Die konkrete Messung bzw. die Abschätzung von Energiebedarfen ist Grundlage der Entwicklung energieeffizienter Softwaresysteme. Zusätzlich werden Techniken, Methoden und Werkzeuge benötigt, die die Entwicklung aktiv unterstützen. Dies reicht von der Spezifikation Energie-bezogener Anforderungen, über die Modellierung von Bedarfen und Verbräuchen, die Verwendung von Entwurfsmustern zur Energie-effizienten Entwicklung, energieeffizienter Programmiersprachen und Compilern bis hin zur Definition und automatischen Erfassung des Energiebedarfs im Rahmen systematischer Testverfahren. Eine zentrale Anforderung ist es hierbei auch, den Software-Entwicklern ein umfassendes Bewusstsein für das *Qualitätskriterium Energieeffizienz* zu vermitteln.

Zu klären sind auch die Eigenschaften von Software-Systemen, die den Energiebedarf bestimmen und wie dies bereits in den frühen Phasen der Entwicklung, aber auch während der Software-Evolution berücksichtigt wird. Ein Ansatz ist hierbei die Nutzung von Zustandsautomaten. Der Hardware-zentrierte Ansatz von (Nieberg et al, 2003) ordnet jeder Komponente eine Menge möglicher Zustände zu und modelliert deren Beziehungen mit Hilfe einer Zustandsübergangsmatrix. Zusätzlich wird jedem Zustand ein Leistungsverbrauch und jedem Zustandswechsel ein Energieverbrauch zugeordnet. Durch Bestimmung der Zeit, in der sich eine Komponente in einem Zustand befindet, wird die Bestimmung des Energieverbrauchs ermöglicht. Eine Weiterentwicklung zur Modellierung der Energiebedarfe von Software sind die sog. Power State Machines (Domis, 2006). Diese nutzen eine Erweiterung der UML zur Spezifikation und betten diese in die KobrA-Methode ein (Atkinson et al, 2007). Damit können Energieaspekte durchgängig in Analyse und Entwurf betrachtet werden.

5 Optimierung des Energiebedarfs von Software

Während hardwareseitig der Energiebedarf eines Systems mittels Schlafmodi, Drosselung der CPU-Leistung, etc. gesteuert und reduziert wird, befindet sich die Forschung zur Optimierung von Software hinsichtlich Energiebedarfs im Anfangsstadium. Erste Arbeiten zeigen, dass der Energiebedarf sowohl beim Entwurf, bei der Implementierung und Weiterentwicklung als auch beim Einsatz von Software berücksichtigt werden muss (Bunse und Höpfner, 2010). Bunse et al (2011) fan-

den heraus, dass unterschiedliche Implementierungsalternativen von Algorithmen auch in ihrem Energieverbrauch variieren. Interessanterweise sind demnach Energieverbrauch und Performanz nicht per se korreliert. Der Grund hierfür ist, dass mittels Software realisierte Algorithmen die zur Verfügung stehende Hardware nutzen und alternative Implementierungen dies auf unterschiedliche Weise tun. Rekursive Algorithmen (z. B. Mergesort) sind deutlich speicherintensiver als nicht rekursive Alternativen (z. B. Insertionsort). Beide Varianten lösen dasselbe Problem (Sortieren einer Liste von Werten). Folglich kann die Nutzung der Ressource Speicher substituiert werden. Speicherzugriffe benötigen deutlich mehr Energie als CPU-Berechnungen (Höpfner und Bunse, 2010b), d. h. mithilfe von Ressourcensubstitution kann Energie eingespart werden (Bunse und Höpfner, 2008). Veijalainen et al (2004) fanden heraus, dass im Vergleich zum einfachen Übertragen von Daten über ein Netzwerk (Ressource Netzwerkkarte) eine vorherige Kompression um einen Faktor größer als 10% und anschließende Dekompression der Daten den Energiebedarf reduziert. Laut Kansal und Zhao (2008) gilt Ähnliches beim Speichern von Daten auf Festplatten.

Somit besteht ein komplexes Beziehungsgeflecht zwischen den verschiedenen Qualitätsfaktoren. Die isolierte Optimierung eines Faktors kann sich negativ auf andere Faktoren auswirken. So haben Experimente (Bunse und Höpfner, 2008) gezeigt, dass die Verbesserung der Performanz eines Systems sich negativ auf Speicher- und Energiebedarf auswirken kann. Optimierung bewegt sich daher stets in einem von den Faktoren aufgespannten Polyeder und die Auswahl der günstigsten Alternative erfordert eine Abwägung der Faktoren gegeneinander.

Die Optimierung des Energie-Bedarfs mit softwaretechnischen Mitteln kann auf verschiedene Art erfolgen (Jelschen et al, 2012). Da der Gesamt-Energiebedarf von IKT-Systemen von den jeweiligen Energiezuständen der einzelnen Hardwarekomponenten abhängt (Shearer, 2007) , kann z. B. durch die Berücksichtigung von Nutzerprofilen, die mittel Software-Instrumentierung ermitteln wurden, mögliches Benutzerverhalten prognostiziert werden und so das Betriebssystem über mögliches Herunterschalten einzelner Komponenten informiert werden (vgl. z. B. (Amur et al, 2012)) . Ist bekannt, dass eine Berechnung nicht schnellst-möglich erfolgen muss, kann hier auch wahlweise, per Strategy-Pattern, auf energetisch günstigere Verfahren umgeschaltet werden. Dieses kann auch den Wechsel zwischen alternativen Hardware-Sensoren (z. B. Positionsberechnung über WLan statt GPS; wenn möglich) umfassen. In ähnlicher Weise können auch in bestimmten Situationen energetisch teure Hardware-Sensoren durch semantisch äquivalente Software-Sensoren ersetzt werden. Software-Reengineering-Techniken stellen Verfahren bereit, die Qualität von Softwaresystemen zu verbessern, ohne deren Funktionalität zu verändern (Chikofsky und Cross, 1990). Die Identifizierung und Beseitigung von energetisch ineffizienten Code-Fragmenten kann mittels Refactoring (Fowler, 1999) erfolgen. (Gottschalk et al, 2012) beschreiben hierzu *Energy Code Smells* sowie einen Graph-basierten Ansatz zur Energie-Effizienz steigernden Code-Transformation.

6 Zertifizierung und Klassifikation

Um energiebewusstere Software am Markt durchsetzen zu können und den Akteuren die Möglichkeit zu geben, verschiedene Softwarelösungen hinsichtlich ihrer Energieeffizienz zu vergleichen, sind valide Kriterien und Zertifizierungen notwendig. Bisher sind diese nur im Bereich Hardware (bspw. TCO, Energy Star, Blauer Engel für Rechenzentren etc.) verfügbar; im Bereich der Software gibt es erste Ansätze zur Bewertung der Ökologie von Websites (bspw. http://www.co2stats.com/ oder (Naumann et al, 2008)) und auch hinsichtlich des Stromverbrauchs von Anwendungen auf Desktop PCs and Servers (Dick et al, 2011) oder der Energieeffizienz von Apps (Wilke et al, 2012a). Grundlagen zur Zertifizierung von Software-Applikationen werden im CoolSoft-Projekt (Wilke et al, 2012b) erarbeitet.

Ein verbreitetes und neutral geprüftes Siegel ist bislang nicht verfügbar. Die Ursache dürfte nicht nur im mangelnden Interesse von Herstellern und Nutzern liegen, sondern vor allem in der bereits beschriebenen Schwierigkeit, den Energieverbrauch wiederholbar zu messen. Software oder auch Softwarekomponenten können nur im Kontext von Standard-Nutzungsverfahren hinsichtlich ihrer Energieeffizienz beurteilt werden. Bei „klassischen“ Algorithmen (bspw. Sortierverfahren) und auch Systemen wie Datenbanken gibt es hierzu etablierte Benchmarking-Ansätze. Bei Anwendungssoftware ist dies schwieriger, zumal zum einen das gleiche Softwareprodukt in verschiedenen Versionen, zum anderen unterschiedliche Software mit gleichem Aufgabentypus verglichen werden kann. Ein Vergleich mit dem Auto zeigt, dass hier bspw. der Literverbrauch pro 100km eine typische Metrik ist, aber dennoch unterschiedliche Fahrzeugklassen im Regelfall nur innerhalb ihrer Gruppe verglichen werden (wer rühmt sich schon damit, dass sein Hybrid-Fahrzeug weniger verbraucht als ein 40-Tonner). Bei Software ist dies aufgrund der zahlreiche Anwendungsgebiete erheblich aufwändiger.

Aus Sicht möglicher Klassifikationen Nachhaltiger Software bietet z. B. das GREENSOFT-Modell einen Referenzrahmen (Naumann et al, 2011). Das GREENSOFT-Modell umfasst vier Teile: Im Software-Lebenszyklus werden nachhaltige Aspekte den gesamten Software-Lebenszyklus betrachtet. Für die Bewertung der Nachhaltigkeit eines Softwareprodukts werden Kriterien und Metriken vorgeschlagen. Ein Vorgehensmodell umfasst Kriterien zur Entwicklung, Anschaffung und Nutzung von Software. Ergänzend werden Handlungsempfehlungen und Werkzeuge zur Unterstützung der unterschiedlichen Akteure integriert.

Für die Analyse und anschließende Energie-Effizienz-Klassifikation kann wieder auf statische und dynamische Verfahren des Software-Reverse-Engineering zurück gegriffen werden (Jelschen et al, 2012). Statische Analysen, die zum Teil auch zur Erkennung von Energy Code Smells genutzt werden, liefern Aussagen über die aktuelle Code-Qualität hinsichtlich des effizienten Energiebedarfs. Dynamische Analysen erlauben die Betrachtung konkreter Nutzungsszenarien, und dem hiermit verbundenen Energiebedarf. Um eine belastbare und auch vergleichende Klassifizierung der Energieeffizienz von Softwaresystemen zu erreichen, ist methodische Unterstützung zur Erstellung umfassender Kriterienkataloge, die mit Reverse-Engineering-Techniken überprüft werden können, zu entwickeln.

7 Zusammenfassung und Ausblick

Energiebedarfe, insbesondere im Bereich der IKT, wachsen kontinuierlich, was aus ökologischen und ökonomischen Gründen Maßnahmen zur Verringerung von Energiebedarfen erfordert. Der Energiebedarf von IKT wird dabei nicht allein durch die zugrundeliegende Hardware bestimmt. Die in diesem Artikel vorgestellten Arbeiten zeigen eine signifikante Korrelation zwischen Energiebedarf einerseits und ausgeführter Software andererseits. Es scheint demnach möglich, durch Adaption von Software den Energiebedarf eines Systems positiv beeinflussen zu können. Grundlage hierfür ist die Existenz eines präzisen Instrumentariums zur Messung der durch Software induzierten Energiebedarfe. Neben „klassischen“ Messverfahren aus der Elektrotechnik gewinnen hier Software-basierte Verfahren an Bedeutung.

Aufbauend auf der Messung von Energiebedarfen zum Baselining und zur Evaluation von Optimierungen stellt insbesondere die Integration des Energiebegriffs in Methoden der Softwaretechnik eine Herausforderung dar. Die Spezifikation von Energiebedarfen in frühen Phasen und die Integration des Energiebegriffs in dynamische Modelle (z./ B./ Zustandsautomaten) sind ein erster Schritt zur ingenieurmäßigen Entwicklung energiebewusster Systeme. Darauf aufbauend können dann Muster zur Kapselung sog. „Best-Practices“, Programmiersprachen und Compiler oder Testverfahren und entwickelt werden.

Parallel zu den vorgenannten Themen ist ein weiterer wichtiger Schwerpunkt die Klassifizierung und Zertifizierung von Energiebedarfen. Analog zu Energieeffizienzklassen wird Nutzern so die Auswahl Energie-effizienter (Software-) Systeme ermöglicht.

Offene Fragen sind dabei in allen Teilgebieten zu finden. Insbesondere sind Punkte wie die Standardisierung von Messansätzen, die Entwicklung von Energie-Benchmarks sowie Verfahren zur Klassifikation (Energie-Label oder Energiekomplexitätsmaße anlog zur Big-O-Notation, Verfahren zur Spezifikation von Energie-Charakteristika sowie die Kapselung von Wissen im Form von Pattern oder Anti-Pattern) und Ansätze zur Selbstadaption sowie Energie-bewussten Compilern bzw. virtuelle Maschinen zu nennen.

Die Workshopreihe „*Energy Aware Software-Engineering and Development*“ zielt auf die intensive Diskussion und Lösung der zuvor skizzierten Herausforderungen der softwaretechnischen Betrachtung der Energieeffizienz von Informationstechnik. Der zweite Workshop[2] findet im Rahmen der BIUS-Tage 2013 in Oldenburg statt und beschäftigt sich in erster Linie mit Verfahren und Techniken zur Messung des Energiebedarfs von Softwaresystemen, Ansätzen zur Definition standardisierter Nutzungsszenarien als Grundlage dynamischer Energiebedarfsmessungen und Ansätzen zur Erstellung ausreichend präziser Energiemodelle als Grundlage für online Messungen.

Danksagung: Die Ausrichtung des 2. Workshops „*Energy Aware Software-Engineering and Development*“ in Oldenburg wäre ohne die großartige Unterstüt-

[2] http://www.se.uni-oldenburg.de/eased2013

zung durch die Organisatoren der BUIS-Tage nicht möglich. Besonderer Dank gebührt daher Jorge Marx Gómez, Barbara Rapp, Andreas Solsbach und dem Team der BUIS-Tage für die große Flexibilität, unseren kleinen Workshop in die BUIS-Tage zu integrieren.

Literatur

Amur H, Nathuji R, Ghosh M, Schwan K, Lee HHS (2012) IdlePower: Application-aware management of processor idle states. In: First Workshop on Managed Many-Core Systems (MMCS, in conjunction with HPDC), Boston, June 2008, URL http://www.cc.gatech.edu/~hamur3/mmcs08_angsl.pdf

Atkinson C, Bostan P, Brenner D, Falcone G, Gutheil M, O H, Juhasz M, Stoll D (2007) Modeling Components and Component-Based Systems in KobrA. In: The Common Component Modeling Example: Comparing Software Component Models. Result from the Dagstuhl research seminar for CoCoME, August 1-3, 2007, LNCS, Springer, Berlin/Heidelberg

Bunse C, Höpfner H (2008) Resource substitution with components — optimizing energy consumption. In: Cordeiro J, Shishkov B, Ranchordas AK, Helfert M (eds) Proc. of the 3rd International Conference on Software and Data Technologie, 5.-8 Juli 2008, Porto, Portugal, INSTICC press, pp 28–35

Bunse C, Höpfner H (2010) Energieeffiziente Software-Systeme. In: Halang WA, Holleczek P (eds) Eingebettete Systeme — Proc. of the real-time 2010 workshop (Echtzeit 2010); 18.-19. November 2010, Boppard. Informatik aktuell, Springer, Berlin/Heidelberg

Bunse C, Höpfner H, Roychoudhury S, Mansour E (2011) Energy efficient data sorting using standard sorting algorithms. In: Cordeiro J, Ranchordas A, Shishkov B (eds) Software and Data Technologies, CCIS, vol 50, Springer, Berlin/Heidelberg, pp 247–260

Chikofsky EJ, Cross JH (1990) Reverse Engineering and Design Recovery: A Taxonomy. IEEE Software pp 13–17

Dick M, Kern E, Drangmeister J, Naumann S, Johann T (2011) Measurement and Rating of Software-induced Energy Consumption of Desktop PCs and Servers. In: Pillmann W, Schade S, Smits P (eds) Innovations in sharing environmental observations and information. Proceedings of the 25th International Conference EnviroInfo October 5 - 7, 2011, Ispra, Italy, pp 290—-299

Domis D (2006) Komponentenbasierte Energiemodellierung am Beispiel eines Ambient Intelligent Systems. Diplomarbeit, TU Kaiserslautern

Fowler M (ed) (1999) Refactoring, Improving the Design of Existing Code. Addison-Wesley

Gottschalk M, Josefiok M, Jelschen J, Winter A (2012) Removing energy code smells with reengineering services. In: Goltz U, Magnor M, Appelrath HJ, Matthies HK, Balke WT, Wolf L (eds) Beitragsband der 42. Jahrestagung der Gesellschaft für Informatik e.V. (GI), Bonner Köllen Verlag, vol 208, pp 441–455

Höpfner H, Bunse C (2010a) Energy Aware Data Management on AVR Micro Controller Based Systems. ACM SIGSOFT SE Notes 35(3)

Höpfner H, Bunse C (2010b) Towards an energy-consumption based complexity classification for resource substitution strategies. In: Balke WT, Lofi C (eds) Proc. of the 22. Workshop on Foundations of Databases, CEUR Workshop Proceeding, vol 581

Höpfner H, Schirmer M, Bunse C (2012) On measuring smartphones' software energy requirements. In: Proceedings of the 7th International Conference on Software Paradigm Trends, ICSOFT 2012, Rome, Italy, July 24-27

Jelschen J, Gottschalk M, Josefiok M, Pitu C, Winter A (2012) Towards applying reengineering services to energy-efficient applications. In: Ferenc R, Mens T, Cleve A (eds) Proceedings of the 16th Conference on Software Maintenance and Reengineering, IEEE

Josefiok M (2012) Energy-Abstraction Layer. Masterthesis, Carl von Ossietzky Universität, Oldenburg

Kansal A, Zhao F (2008) Fine-grained energy profiling for power-aware application design. ACM SIGMETRICS Performance Evaluation Review (36):26–31

Naumann D, Dick M, Kern E, Johann T (2011) The GREENSOFT Model: A Reference Model for Green and Sustainable Software and its Engineering. SUSCOM (Sustainable Computing: Informatics and Systems) (1-4):294—-304

Naumann S, S Gresk S, Schäfer K (2008) How Green is the Web? Visualizing the Power Quality of Websites. In: Möller A, Page B, Schreiber M (eds) EnviroInfo 2008. Environmental Informatics and Industrial Ecology, 22nd International Conference on Environmental Informatics. Aachen, pp 62–65

Nieberg T, Dulman S, Havinga P, van Hoesel L, Wu J (2003) Ambient Intelligence. Impact on Embedded System Design Ambient Intelligence: Collaborative Algorithms for Communication in Wireless Sensor Networks. Kluwer Academic Punblishers, Dordrecht

Shearer F (ed) (2007) Power Management in Mobile Devices . Newnes

Veijalainen J, Ojanen E, Haq MA, Vahteala VP, Matsumoto M (2004) Energy Consumption Tradeoffs for Compressed Wireless Data at a Mobile Terminal. IEICE Transactions on Communications E87-B:1123–1130

Wilke C, Richly S, Piechnick C, Götz S, Püschel G, Aßmann U (2012a) Comparing Mobile Applications' Energy Consumption. Tech. Rep. TUD-Fl12-10, Technische Universität Dresden

Wilke C, Richly S, Püschel G, Piechnick C, Götz S, mann UA (2012b) Energy labels for mobile applications. In: Proceedings des 1. Workshop zur Entwicklung energiebewusster Software (EEbS 2012)