DOKUMENTATIONEN

07/2015

Nachhaltige Software

Dokumentation des Fachgesprächs „Nachhaltige Software“ am 28.11.2014

Für Mensch & Umwelt

Umwelt Bundesamt

DOKUMENTATIONEN 07/2015

Umweltforschungsplan des
Bundesministeriums für Umwelt,
Naturschutz, Bau und Reaktorsicherheit

Forschungskennzahl 3712 95 301

Nachhaltige Software

Dokumentation des Fachgesprächs „Nachhaltige Software“ am 28.11.2014

Durchgeführt im Rahmen des Forschungsvorhabens „Ökologische und ökonomische Aspekte beim Vergleich von Arbeitsplatzcomputern für den Einsatz in Behörden unter Einbeziehung des Nutzerverhaltens“

Zusammengestellt von

Jens Gröger
Öko-Institut e.V., Berlin

Marina Köhn
Beratungsstelle nachhaltige Informations- und Kommunikationstechnik (Green-IT), Umweltbundesamt, Dessau

mit Beiträgen von

Erik Albers
Free Software Foundation Europe e.V., Berlin

Jens Gröger
Öko-Institut e.V., Berlin

Patrik Löhr
Posteo e.K., Berlin

Wolfgang Lohmann
Consultant, St. Gallen

Stefan Naumann
Institut für Softwaresysteme in Wirtschaft, Umwelt und Verwaltung, Hochschule Trier, Trier

Im Auftrag des Umweltbundesamtes

Impressum

Herausgeber:
Umweltbundesamt
Wörlitzer Platz 1
06844 Dessau-Roßlau
Tel: +49 340-2103-0
Fax: +49 340-2103-2285
info@umweltbundesamt.de
Internet: www.umweltbundesamt.de

/umweltbundesamt.de
/umweltbundesamt

Durchführung der Studie:
Öko-Institut e.V.
Merzhauser Str. 173
79100 Freiburg

Abschlussdatum:
31.12.2014

Redaktion:
Beratungsstelle nachhaltige Informations- und Kommunikationstechnik (Green-IT)
Marina Köhn

Publikationen als pdf:
http://www.umweltbundesamt.de/publikationen/nachhaltige-software

ISSN 2199-6571

Dessau-Roßlau, Juni 2015

Die vorliegende Dokumentation wurde im Rahmen des Umweltforschungsplans im Auftrag des Umweltbundesamtes erarbeitet. Die Beiträge der Autoren müssen nicht in allen Punkten die Auffassungen des Auftraggebers wiedergeben.

Kurzbeschreibung

Für Nachhaltige Software gibt es bislang keine einheitliche Definition oder allgemein akzeptierte Standards. Das Umweltbundesamt und das Öko-Institut e.V. veranstalteten daher am 28. November 2014 ein Fachgespräch mit Vertreterinnen und Vertretern aus Wissenschaft, Softwareentwicklung und Anwendung, um ein gemeinsames Verständnis für Nachhaltige Software zu entwickeln. In der vorliegenden Dokumentation werden die Kurzvorträge des Fachgesprächs zusammengefasst und die geführten Diskussionen dokumentiert. Weiterhin wird ein Ausblick darauf gegeben, welche weiteren Forschungsaktivitäten notwendig sind, um Nachhaltige Software zu entwickeln und perspektivisch zu kennzeichnen.

Abstract

So far, no consistent definitions or generally accepted standards exist for Sustainable Software. On 28th November 2014, the Federal Environment Agency and Oeko-Institut e.V. hosted a consultation, inviting representatives from science, software development and application to develop a common understanding of Sustainable Software. This documentation summarizes the short presentations and keeps records of the experts' discussions. Additionally, an outlook is given on further research that is needed to develop Sustainable Software and for its future labelling.

Inhaltsverzeichnis

Einleitung 9
1 Software im Kontext der nachhaltigen Produktpolitik 10
1.1 Instrumente der Produktpolitik 10
1.1.1 Hintergrund zur Entstehung der integrierten Produktpolitik 10
1.1.2 Umsetzung der Instrumente in aktueller Produktpolitik 10
1.2 Umweltzeichen als freiwilliges Informationsinstrument 11
1.3 Entwicklung von Vergabekriterien für Umweltzeichen 12
1.3.1 Produktnachhaltigkeitsanalyse 12
1.3.2 Beispielhafte Vergabekriterien 14
1.4 Übertragbarkeit auf Software-Produkte 14
1.5 Software als Gegenstand der Produktpolitik 15
1.6 Quellenverzeichnis 16
2 Green Software Engineering - Modelle und Perspektiven 18
2.1 Vorbemerkung 18
2.2 Was ist Green Software Engineering? 18
2.3 Messung des Energieverbrauchs von Software 20
2.4 Ausblicke und Forschungsansätze 21
2.5 Weiterführende Informationen 22
3 Einfluss von Software auf Energie- und Ressourceneffizienz 23
3.1 Einführung 23
3.2 Beschreibung von Softwaresystemen 23
3.3 Von der Software zum Energieverbrauch durch Hardware 23
3.4 Direkter durch Software verursachter Energieverbrauch 24
3.5 Umweltauswirkungen durch erforderliche Hardware 25
3.6 Nachhaltigkeit durch IKT 25
3.7 Immaterieller Charakter der Software 26
3.8 Schlussfolgerungen 27
3.9 Quellenverzeichnis 28
4 Software im Kontext der Nachhaltigkeit 29
4.1 Zusammenfassung 29
4.2 Zum Begriff „Nachhaltigkeit“ 29
4.3 Nachhaltigkeit digitaler Ressourcen 30
4.4 Nachhaltigkeit 'durch', aber auch 'von' Software 31

4.5 Freie Software – Definition, Entstehung und heutige Bedeutung 32
4.6 Freie Software und ökologische Nachhaltigkeit 33
4.7 Modularität, Freie Software und Effizienz 34
4.8 Hardware als Ressource verstehen 34
4.9 Gemeinsame Lösungen für eine gemeinsame Ressource 35
4.10 Ausblick 36
4.11 Quellenverzeichnis 37
5 Nachhaltiger Betrieb von Online-Diensten 38
5.1 Eingangsstatement 38
5.2 Nachhaltigkeitskonzept 38
5.3 Motivation 38
5.4 Nutzung von Ökostrom 39
5.5 Hardware 39
5.6 Beispiele für nachhaltigere Software 39
5.7 Nachhaltigkeit vs. Sicherheit 41
5.8 Empfehlungen 41
6 Diskussionsbeiträge zu Nachhaltiger Software 42
6.1 Was ist „Nachhaltige Software“? 42
6.2 Kriterien für „Nachhaltige Software“? 43
6.3 Systemgrenzen von „Nachhaltiger Software“? 45
6.4 Welche Standards und Methoden für „Nachhaltige“ bzw. „Green Software“ gibt es bereits? 45
7 Zusammenfassung und Ausblick 47

Einleitung

Wird von Green IT (Grüner Informationstechnik) gesprochen, so denkt man dabei meist an energieeffiziente und ressourcenschonende Hardware. Von ebenso großer Bedeutung für die Nachhaltigkeit eines IT-Systems ist jedoch die Software. Wachsende Datenmengen und immer komplexere Programme erfordern den beständigen Ausbau von IT-Netzen, Datenspeichern und Rechenkapazitäten.

Um die Bedeutung von Software für die Nachhaltigkeit von IT-Systemen herauszuarbeiten, veranstalteten Umweltbundesamt und Öko-Institut e.V. am 28. November 2014 ein Fachgespräch zu „Nachhaltiger Software“. Zusammen mit Vertreterinnen und Vertretern aus Wissenschaft, Softwareentwicklung und Anwendung wurden Ansatzpunkte für eine umwelt- und sozialverträgliche Software aus unterschiedlichen Perspektiven diskutiert.

Unter „Nachhaltiger Software“ wird (im weitesten Sinne) Software verstanden, die entlang ihres Lebensweges (Entwicklung, Anwendung, Außerbetriebnahme) die Umwelt wenig belastet und somit ressourcenschonend und energieeffizient ist. Dies kann beispielsweise dadurch realisiert sein, dass sie effizient und schlank programmiert sowie modular und erweiterbar ist oder langfristig gepflegt wird. Weiterhin kann sich die Software dadurch auszeichnen, dass sie in ihrer Anwendung einen positiven Effekt auf die Umwelt hat, beispielsweise indem sie hilft, Energie und Ressourcen einzusparen oder Prozesse zu optimieren.

In der vorliegenden Dokumentation werden die Kurzvorträge des Fachgesprächs zusammengefasst und die geführten Diskussionen dokumentiert. Weiterhin wird ein Ausblick darauf gegeben, welche weiteren Forschungsaktivitäten notwendig sind, um Nachhaltige Software zu entwickeln und perspektivisch mit einem Label zu kennzeichnen. Als mögliches Kennzeichnungssystem wird dabei das Umweltzeichen „Blauer Engel“ vorgeschlagen.

1 Software im Kontext der nachhaltigen Produktpolitik

Autor: Dipl.-Ing. Jens Gröger, Senior Researcher, Bereich Produkte & Stoffströme, Öko-Institut e.V., Schicklerstraße 5-7, 10179 Berlin, j.groeger@oeko.de

1.1 Instrumente der Produktpolitik

1.1.1 Hintergrund zur Entstehung der integrierten Produktpolitik

Bei der klassischen Umweltpolitik, die sich in Deutschland in den siebziger Jahren etablierte, lag der Fokus auf der Begrenzung von Schadstoffemissionen von Fabriken und Kraftwerken (z. B. Bundes-Immissionsschutzgesetz) sowie auf dem Naturschutz (z. B. Umweltverträglichkeitsprüfung). Welche Eigenschaften die hergestellten Produkte hatten, lag weitestgehend in der Verantwortung der Unternehmer selbst und wurde allenfalls durch Normen oder Industriestandards festgelegt. Die in den achtziger Jahren voranschreitende Zerstörung der Ozonschicht durch Fluorchlorkohlenwasserstoffe (FCKWs) führte schließlich zu einer stärkeren Wahrnehmung von Produkten als Verursacher von Umweltbelastung. Die FCKW-Halon-Verbots-Verordnung aus dem Jahr 1991 reagierte darauf mit einem Verbot von besonders umweltrelevanten FCKWs in Spraydosen, Kältemitteln und Schaumstoffen. Seither sind Produkte immer weiter in den Regelungsbereich der Umweltpolitik gerückt und das Politikfeld der nachhaltigen Produktpolitik entstand.

In den nuller Jahren wurde durch die Europäische Kommission mit einem Grünbuch zur Integrierten Produktpolitik (EU 2001) der politische Grundstein einer auf Aspekte der Nachhaltigkeit ausgerichtete Produktpolitik gelegt. Der als „Integrierte Produktpolitik“ bezeichnete Ansatz nimmt dabei den gesamten Lebenszyklus eines Produktes in den Blick und begrenzt auf der einen Seite die negativen Umweltwirkungen von Produkten und fördert auf der anderen Seite umweltfreundliche Produkte und ökologische Innovationen (Rubik et al. 2005).

Das Grünbuch zur Integrierten Produktpolitik (EU 2001) schlägt folgende Instrumente vor, mit denen nachhaltige Produkte gefördert werden können:

- Nutzung von finanziellen Mechanismen: Differenzierte Besteuerung von Produkten in Abhängigkeit von ihren Umwelteigenschaften, Ausweitung der Produzentenverantwortung, Umwelthaftung, staatliche Finanzhilfen für umweltfreundliche Produkte.
- Förderung der Nachfrage nach umweltfreundlichen Produkten: Umweltkennzeichnung, Produktinformationen, Verbesserung von Informationsangeboten, Beschaffung umweltfreundlicher Produkte durch die öffentliche Hand.
- Stärkung der Vorreiterrolle der Wirtschaft: Befähigung zur Ökobilanzierung / Umweltproduktdeklaration, Anforderungen an die umweltgerechte Produktentwicklung (Ökodesign), Normung mit Blick auf ökologische Aspekte, Durchführung von Pilotprojekten.
- Ergänzende Instrumente: Einführung von Umweltmanagementsystemen, Förderung von Forschung & Entwicklung, Projektförderung, Umweltberichterstattung.

1.1.2 Umsetzung der Instrumente in aktueller Produktpolitik

Sieht man sich die aktuelle Produktpolitik an, so sind viele der durch die EU-Kommission vorgeschlagenen Instrumente zwischenzeitlich realisiert oder in politische Maßnahmen umgesetzt. Im Jahr 2005 wurde mit der Rahmenrichtlinie für die Festlegung von Anforderungen an die umweltgerechte Gestaltung energiebetriebener Produkte (EU 2005) der Grundstein für Ökodesign-Anforderungen gelegt. Die Instrumente der Energieeffizienzkennzeichnung (EU 1992b, EU 2010b) wurden weiter verstetigt und das EU-Umweltzeichen (EU 1992a, EU 2010a) wurde durch die Einbeziehung weiterer Produktgruppen weiter ausgebaut.

Die verschiedenen Instrumente setzen bei unterschiedlichen Anforderungsniveaus (Umweltverträglichkeit bzw. Nachhaltigkeit der Produkte) an.

Eingriffe in den Markt: Mit den Mindestanforderungen der Europäischen Richtlinien RoHS (Restriction of Hazardous Substances), ELV (End of Life Vehicles), Ökodesign (EU 2009) und REACH (Registration, Evaluation, Authorisation and Restriction of Chemicals) werden Anforderungen festgelegt, die von Produkten eingehalten werden müssen, um auf dem Europäischen Markt verkauft werden zu dürfen. Nur Produkte, die diese Verordnungen zu Stoffverboten und Mindesteffizienzen einhalten, dürfen das CE-Kennzeichen (CE – Communauté Européenne) tragen und damit ihre Konformität zu den in der EU geltenden Anforderungen deklarieren.

Markttransformation: Als Informationsangebot für Konsumentinnen und Konsumenten wurde die Energieverbrauchskennzeichnung eingeführt, die energieverbrauchende Haushaltsgeräte und Fahrzeuge mit einem leicht verständlichen Label und der Kennzeichnung A bzw. A+++ (sehr energieeffizient) bis G (sehr ineffizient) kennzeichnet. Weitere Angaben auf der Kennzeichnung können beispielsweise Wasserverbrauch, Geräuschemissionen oder Verbrauchswerte pro Nutzungseinheit (Jahr, Waschgang, Fahrleistung usw.) sein. Mit der Pflichtkennzeichnung, die an Produkten bei den Verkaufsstellen angebracht werden muss, soll der Markt dahingehend beeinflusst werden, dass solche Produkte bevorzugt gekauft werden, die geringe Umweltwirkungen und einen geringen Energieverbrauch aufweisen.

Förderung von Innovation: Am oberen Ende des Anforderungsniveaus setzen die Instrumente umweltfreundliche Beschaffung (GPP – Green Public Procurement) und Umweltzeichen an. Die öffentliche Hand kauft bei der umweltfreundlichen Beschaffung bevorzugt sparsame und umweltfreundliche Produkte ein. Sie nimmt dabei einen möglicherweise höheren Anschaffungspreis in Kauf, um gezielt aus ökologischer Sicht vorteilhafte Produkte zu fördern. Für private Konsumentinnen und Konsumenten bieten produktbezogene Umweltzeichen (z. B. Blauer Engel, EU-Umweltzeichen) eine Orientierung. Umweltzeichen (nach ISO 14024) kennzeichnen ökologische Spitzenprodukte nach einem lebenszyklusbezogenen Ansatz und bieten den Herstellern die Möglichkeit, ihre umweltfreundlichen Produkte gegenüber der Konkurrenz hervorzuheben und zeigen dem Kunden „rundum gute“ Produkte an.

1.2 Umweltzeichen als freiwilliges Informationsinstrument

Hersteller von umweltfreundlichen Produkten und Anbieter von umweltfreundlichen Dienstleistungen können zur Darstellung ihres Umweltengagements und der Unterstützung ihrer Glaubwürdigkeit Umweltzeichen nutzen. Dabei gelten solche Umweltzeichen als besonders glaubwürdig, die von einer unabhängigen Stelle vergeben werden. Umweltzeichen werden direkt am Produkt oder im Produktkatalog angebracht und sollen den Käufer bei seiner Kaufentscheidung unterstützen. Um eine schnelle Entscheidung zu ermöglichen, zeichnen sich Umweltzeichen durch ein wiedererkennbares Wort- oder Bildzeichen (Schriftzug oder Logo) aus. Durch das Zeichen oder den begleitenden Text wird die besondere Umwelteigenschaft des Produktes hervorgehoben (z. B. „energiesparend“). Adressaten von Umweltzeichen sind private Endverbraucher, aber auch professionelle Einkäufer in Unternehmen und bei der öffentlichen Hand. Umweltzeichen oder die dahinter liegenden Kriterien werden dort genutzt, um besonders umweltfreundliche Produkte zu beschaffen und die Nachhaltigkeitsziele der Unternehmen bzw. der öffentlichen Hand umzusetzen. Das Anforderungsniveau, das mit Umweltzeichen gekennzeichnete Produkte erfüllen, beschreibt darüber hinaus auch die (aus Umweltgesichtspunkten) beste am Markt verfügbare Technologie (best available technology). Dieses Niveau wird bei der Festlegung von gesetzlichen Mindeststandards (z. B. Ökodesign-Anforderungen) daher oft auch als zu erreichende Zielgröße angesetzt. Umweltzeichen haben dadurch eine Lenkungswirkung, die weit über das jeweilige Produkt hinausgeht.

Für die Umweltzeichen gibt es internationale Standards, die festlegen, wie die dahinter liegenden Kriterien entwickelt werden und welche organisatorischen Voraussetzungen für eine geordnete Zeichenvergabe geschaffen werden müssen. Besonders hohe Standards gelten für „Zertifizierte Umweltkennzeichnungen“, so genannte Typ-I-Umweltzeichen nach der Norm EN ISO 14024. Bekannte Ökolabel dieses Typs sind der Blaue Engel, das EU-Ecolabel oder das Österreichische Umweltzeichen (vgl. Abbildung 1). Bei Umweltzeichen, die nach der Norm entwickelt wurden, handelt es sich um freiwillige Zeichen, die allen interessierten Anbietern des jeweiligen Produktes (oder der jeweiligen Dienstleistung) offen stehen. Die Kriterien müssen in einem transparenten und wissenschaftlichen Prozess durch die Untersuchung des gesamten Lebenszyklus des Produktes erarbeitet werden. Die Erfüllung der Kriterien muss durch den Zeichennutzer durch unabhängige Prüfinstitute nachgewiesen werden. Die Zeichenvergabe erfolgt durch einen unabhängigen Dritten, der vom Zeichennutzer finanziell nicht abhängig ist.

Abbildung 1: Beispiele für Umweltzeichen

Zusammenstellung: Öko-Institut. Die Logos sind eingetragene Marken der jeweiligen Zeicheninhaber

1.3 Entwicklung von Vergabekriterien für Umweltzeichen

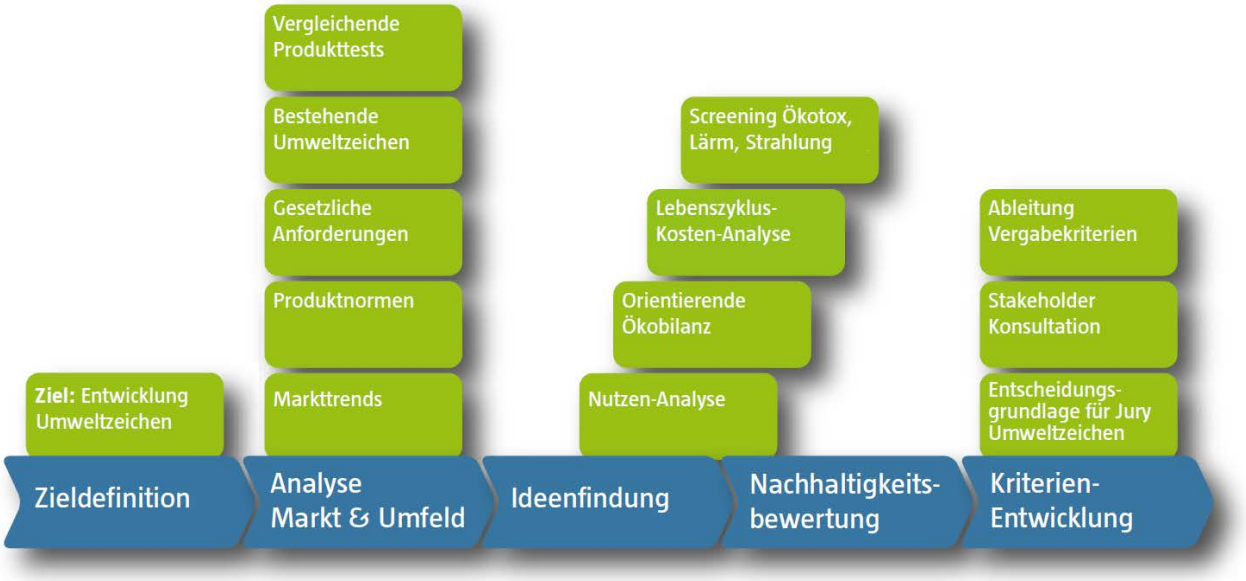
1.3.1 Produktnachhaltigkeitsanalyse

Das Öko-Institut hat mit PROSA (Product Sustainability Assessment)[1] eine Methode entwickelt, Produkte und deren Herstellungsprozesse auf Nachhaltigkeitsaspekte hin zu untersuchen, strategisch zu bewerten und – je nach Fragestellung – daraus Maßnahmen abzuleiten. Die Methode eignet sich auch für die Entwicklung von Vergabekriterien für Typ-I-Umweltzeichen.

In Abbildung 2 wird der schematische Ablauf der Kriterienentwicklung mit PROSA dargestellt.

[1] www.prosa.org

Abbildung 2: Ablaufschema PROSA (Product Sustainability Assessment)



Quelle: Öko-Institut

Die **Zieldefinition** beinhaltet die Festlegung, für welches Produkt Umweltzeichenkriterien entwickelt werden sollen. Diese Festlegung ist nicht immer trivial, weil der gleiche Produktnutzen durch ganz unterschiedliche technische Varianten realisiert werden kann (beispielsweise kann ein Videokonferenzsystem eine reine Software-Lösung für ein bestehendes Computersystem mit Webcam sein oder eine eigenständige Hardware-Lösung mit automatisch schwenkbarer Videokamera und Videomonitor).

In der **Marktanalyse** wird eine Übersicht über bestehende Produkte und deren technischen Eigenschaften erstellt. Hierzu tragen vergleichende Produkttest (z. B. von Verbraucherschutzorganisationen) bei. Weiterhin wird anhand von Markttrends untersucht, wie sich die Produkte zukünftig weiter entwickeln werden. Die **Umfeldanalyse** ordnet die Produktgruppe in bestehende Umweltzeichensysteme, gesetzliche Anforderungen und Normen ein.

Der Kern der Produktnachhaltigkeitsanalyse ist die Untersuchung der ökologischen, sozialen und ökonomischen Wirkungen eines Produktes entlang seines **Lebenszyklus**, also von der Rohstoffgewinnung über den Transport, die Verarbeitung, die Verpackung, die Gebrauchsphase des Produktes bis zur Entsorgung. Hierzu werden orientierende Ökobilanzen (Umweltaspekte), Lebenszykluskosten-Analysen (ökonomische Aspekte) sowie Methoden zur Bestimmung von Gesundheitsbelastungen (soziale Aspekte) und des Produkt-Nutzens angewandt.

Hierdurch lassen sich relevante Hot-Spots identifizieren. Das sind solche Produktwirkungen, die einen besonders hohen (in der Regel negativen) Einfluss auf die Nachhaltigkeit eines Produktes haben. Die Vergabekriterien eines Umweltzeichens setzen bei den Hot-Spots an, um eine hohe Umweltentlastung zu erreichen. So wurde beispielsweise bei der Produktnachhaltigkeitsanalyse von Smartphones festgestellt, dass der Energieverbrauch der Geräte selbst kaum zu deren Umweltbelastung beiträgt, sondern vielmehr der Herstellungsaufwand der Geräte (Manhart 2012). Die Kriterien eines Umweltzeichens müssen in diesem Fall darauf ausgerichtet sein, die Produktlebensdauer durch geeignete Anforderungen zu verlängern (Update-Fähigkeit, Reparierbarkeit, Qualität der Akkus).

Die **Kriterienentwicklung** wird durch einen Stakeholder-Prozess begleitet, bei dem interessierte Kreise (Hersteller, Umwelt- und Verbraucherverbände, Vertreter aus Wissenschaft und Forschung, Politik usw.) innerhalb von Fachgesprächen, Expertenanhörungen und Konsultationen zur Umsetz-

barkeit und dem Ambitionsniveau der Vergabekriterien befragt werden. Das Umweltzeichen selbst wird im Fall des Blauen Engels von der *Jury Umweltzeichen* verabschiedet, einem Gremium aus gesellschaftlich relevanten Gruppen, das das oberste Beschlussorgan des Blauen Engels darstellt.

1.3.2 Beispielhafte Vergabekriterien

Umweltzeichen signalisieren anhand ihrer Umschrift (z. B. „energieeffizient“) oder ihrem Schutzziel (z. B. „schützt das Klima“), dass durch sie eine spezielle, negative Umweltwirkung reduziert wird. Bei Typ-I-Umweltzeichen, wie dem Blauen Engel, ist jedoch in der Regel mehr als nur eine Umweltwirkung die Voraussetzung dafür, dass ein Zeichen genutzt werden darf („multikriterieller Ansatz“).

Elektro- und Elektronikgeräte, die mit dem Umweltzeichen Blauer Engel mit dem Schutzziel „schützt das Klima“ gekennzeichnet sind, erfüllen beispielsweise folgende Kriterien:

- Begrenzung des Energieverbrauchs (z. B. Bereitschaft, Betrieb)
- Reduktion von umwelt- oder gesundheitsgefährdenden Stoffe im Produkt oder bei der Herstellung
- Ressourcenschonender Materialeinsatz (z. B. Nutzung von Recyclingmaterialien)
- Langlebigkeit (Reparierbarkeit, Ersatzteilversorgung, Update-Fähigkeit)
- Recyclinggerechte Konstruktion (z. B. lösbare Verbindungen, geringe Materialvielfalt, Kennzeichnung von Kunststoffen)
- Geringe Emissionen elektromagnetischer Strahlen
- Geringe Geräuschemissionen
- Höhe der CO_2-Emissionen bei Transport und Nutzung

1.4 Übertragbarkeit auf Software-Produkte

Um die Kriterienentwicklung für ein Umweltzeichen auf Software-Produkte zu übertragen und damit „Nachhaltige Software“ auszeichnen zu können, ist eine Reihe von methodischen Herausforderungen zu überwinden.

Die erste Herausforderung besteht darin, die **Software als eindeutiges Produkt** zu fassen und damit festzulegen, welches Produkt untersucht wird bzw. die Berechtigung dafür erhält, mit einem Label gekennzeichnet zu werden. Ist beispielsweise die Nutzung eines Online-Dienstes ein Produkt im Sinne eines Umweltzeichens oder nur solche Software, die als lauffähige Datei auf einem Datenträger abgelegt werden kann? Selbst bei Software, die auf stationären Computern installiert ist, ändert sich der Code und damit die Eigenschaften der Software bei jedem Update und unterscheidet sich je nach Betriebssystem oder Hardware-Umgebung. Eine Annäherung an diese Problematik ist die Beschreibung der Software als zu erfüllende Funktion/ zu bedienender Produktnutzen, die für die zu kennzeichnende Software konstant bleiben müssen.

Die nächste Herausforderung ist die **Identifikation der maßgeblichen Wirkungen** einer Software. Grundsätzlich lässt sich die Systematik von Lebenszyklusphasen auch auf Software übertragen. Die Lebenszyklusphasen von Software lassen sich, vergleichbar mit einem physikalischen Produkt, wie folgt fassen:

- **Herstellung**: Konzeption, Umsetzungsdesign, Programmierung, Testphase, Dokumentation
- **Transport**: Distribution, Datenaustausch zum Hersteller
- **Inbetriebnahme**: Installation, Integration, Datenübernahme von Vorläufersoftware
- **Gebrauchsphase**: Nutzung der Software beim Kunden, Instandhaltung, Updates
- **Entsorgung**: Deinstallation, Datenübergabe an Folgesoftware, Datenlöschung

Die maßgeblichen Wirkungen auf die unterschiedlichen Nachhaltigkeitsdimensionen lassen sich analog den Lebenszyklusphasen zuordnen:

- Hard- und Software-Ressourcen während der Herstellung (Energie, Entwicklungssysteme, Personal, Infrastruktur)
- Hardware- und Software-Ressourcen bei der Nutzung (Rechenleistung CPU – central processing unit und GPU – graphics processing unit, RAM – random access memory, Festplattenplatz, Lebensdauer, Übertragungswege, abhängige Software)
- Energie- und Rohstoffverbrauch bei der Nutzung
- Einsparung von Energie- und Reduzierung von Ressourceninanspruchnahme durch die Nutzung
- Personeller/ökonomischer Aufwand für Einarbeitung, Schulung, Ausfallzeit, Aufwand zur Pflege, Wartung und Instandhaltung
- Ökonomischer Aufwand: Investition für Anschaffung, vertragliche Verpflichtungen, Service
- Entsorgungsaufwand (Deinstallation)

Die möglicherweise größte Herausforderung ist es, geeignete **Mindestkriterien** zu entwickeln, die durch Nachhaltige Software eingehalten werden müssen. Zu Mindestkriterien gehören in der Regel immer auch geeignete Nachweisverfahren und Prüfmethoden. Wird beispielsweise ein maximaler Energieverbrauch für Software festgeschrieben, so muss gleichzeitig ein reproduzierbarer Prüfaufbau festgelegt werden und im Detail bestimmt werden, auf welcher Hardwareplattform die jeweilige Software ausgeführt wird und welche Operationen die Software während der Messungen ausführt. In der Praxis wird dies wegen der Vielfalt an unterschiedlichen Software-Produkten, Hardware-Umgebungen und Konfigurationen nur sehr eingeschränkt möglich sein.

Statt fester Mindestkriterien, die in Summe eingehalten werden müssen, könnten die Kriterien zur Vergabe eines Umweltzeichens für Nachhaltige Software als **Katalog von Nachhaltigkeitseigenschaften** formuliert werden. Die Nachhaltigkeitseigenschaften beschreiben positive Aspekte, die durch die Software abgedeckt werden. Als nachhaltig kann dann Software bezeichnet werden, die eine Mindestanzahl an Nachhaltigkeitsaspekten erfüllt. Ein ähnliches Vorgehen wählt die Vergabegrundlage des Blauen Engels für umweltschonenden Schiffsbetrieb (RAL 2010). Dort sind rund 60 verbindliche und optionale Einzelkriterien aufgeführt. Um das Umweltzeichen zu erhalten, muss ein Mindestmaß an Kriterien erfüllt werden, jedoch nicht alle gleichzeitig.

Auch wenn die Entwicklung eines Umweltzeichens für Nachhaltige Software auf den ersten Blick nicht trivial scheint, so gibt es dennoch genügend Ansatzpunkte, Vergabekriterien entsprechend der Norm ISO 14024 abzuleiten. Die Entwicklung der Kriterien sollte dabei nicht auf einzelne Umwelt- bzw. Nachhaltigkeitsaspekte von Software beschränkt sein (wie beispielsweise den durch Software verursachten Energieverbrauch), sondern nach Möglichkeit eine große Bandbreite an Nachhaltigkeitskriterien abdecken.

1.5 Software als Gegenstand der Produktpolitik

Bislang wird Software durch die Instrumente der Produktpolitik wenig beachtet. Es ist jedoch absehbar, dass sich dies zukünftig ändern wird. Es werden seitens der Politik sehr hohe Erwartungen in die Möglichkeiten von Software gesteckt, Energieverbräuche zu reduzieren und Wirtschaftsabläufe zu optimieren. Stichworte hierzu sind beispielsweise

- Smart Metering: Energieverbrauchsmessung mit dem Ziel, Verbrauch und Kosten zu senken,
- Smart Grids: intelligente Stromnetze, die Energieverbraucher und Energieerzeuger vernetzen und gegenseitig steuern, mit dem Ziel, Überkapazitäten und Engpässe zu reduzieren sowie
- Autonomes Fahren: selbstlenkende Fahrzeuge, die der Automobilindustrie einen Innovationsvorteil verschaffen und zugleich eine höhere Auslastung des vorhandenen Straßennetzes zulassen.

Als konkrete Maßnahme zur Förderung von Software, die dazu beiträgt, den Energieverbrauch von Unternehmen zu senken, hat die Bundesregierung eine „Richtlinie für die Förderung von Energiema-

nagementsystemen“ erlassen (Bundesregierung 2013). Unternehmen mit Sitz oder Niederlassung in der Bundesrepublik Deutschland erhalten nach der Richtlinie einen Zuschuss von 20% für den Erwerb von Software für Energiemanagementsysteme.

Weniger konkret sind die Ankündigungen, die die Bundesregierung in ihrer Digitalen Agenda (Bundesregierung 2014) macht. In dieser Agenda wird beschrieben, wie die Digitalisierung genutzt werden soll, „um Deutschlands Rolle als innovative und leistungsstarke Volkswirtschaft in der Europäischen Union und der Welt auszubauen“. Die Digitale Agenda kündigt unter anderem an, dass die Hemmnisse für die Nutzung von Open Source Software in Bundesverwaltungen abgebaut werden sollen. Die (heimische) Softwareentwicklung soll befördert werden, indem Kompetenzen aufgebaut werden und Unternehmen und Behörden befähigt werden, Softwarekomponenten technologisch zu beherrschen. Weiterhin möchte sich die Bundesregierung stärker dem Datenschutz widmen und sicherstellen, dass persönliche Daten besser geschützt werden und Geschäftsgeheimnisse vertraulich bleiben.

Eine freiwillige Produktkennzeichnung, beispielsweise durch ein Umweltzeichen oder ein Nachhaltigkeitslabel, kann als Einstieg in eine gezieltere Produktpolitik für Software dienen. Indem Software-Produkte gekennzeichnet werden, die einen Umweltvorteil bieten oder einen Beitrag zur Nachhaltigkeit leisten, kann sich Nachhaltige Software besser am Markt etablieren. Für private und professionelle Einkäufer bzw. Nutzer von Software könnte ein solches Kennzeichen Orientierung bieten und für Software-Unternehmen könnten die Kriterien eine Richtlinie zur Softwareentwicklung darstellen.

1.6 Quellenverzeichnis

Bundesregierung (2013): Bundesministerium für Wirtschaft und Technologie; Richtlinie für die Förderung von Energiemanagementsystemen; 22. Juli 2013; http://www.bmwi.de/BMWi/Redaktion/PDF/P-R/richtlinie-energiemanagementsysteme; aufgerufen am 19.11.2014.

Bundesregierung (2014): Bundesministerium für Wirtschaft und Energie, Bundesministerium des Innern, Bundesministerium für Verkehr und digitale Infrastruktur; August 2014; http://www.bmwi.de/BMWi/Redaktion/PDF/Publikationen/digitale-agenda-2014-2017; aufgerufen am 19.11.2014.

EN ISO 14024: Umweltkennzeichnungen und -deklarationen (Umweltkennzeichnung Typ I) - Grundsätze und Verfahren (ISO 14024:1999); Deutsche Fassung EN ISO 14024:2000.

EU (1992a): Verordnung (EWG) Nr. 880/92 des Rates vom 23. März 1992 betreffend ein gemeinschaftliches System zur Vergabe eines Umweltzeichens, http://eur-lex.europa.eu/legal-content/DE/TXT/PDF/?uri=CELEX:31992R0880&qid=1421143395502&from=DE .

EU (1992b): Richtlinie 92/75/EWG des Rates vom 22. September 1992 über die Angabe des Verbrauchs an Energie und anderen Ressourcen durch Haushaltsgeräte mittels einheitlicher Etiketten und Produktinformationen, http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31992L0075:DE:PDF .

EU (2001): Grünbuch zur Integrierten Produktpolitik, KOM (2001) 68 endgültig, 07.02.2001, Brüssel, http://eur-lex.europa.eu/legal-content/DE/TXT/PDF/?uri=CELEX:52001DC0068&qid=1420459357800&from=DE, aufgerufen am 19.11.2014.

EU (2005): Richtlinie 2005/32/EG des Europäischen Parlaments und des Rates vom 6. Juli 2005 zur Schaffung eines Rahmens für die Festlegung von Anforderungen an die umweltgerechte Gestaltung energiebetriebener Produkte und zur Änderung der Richtlinie 92/42/EWG des Rates sowie der Richtlinien 96/57/EG und 2000/55/EG des Europäischen Parlaments und des Rates, http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2005:191:0029:0058:DE:PDF .

EU (2009): Richtlinie 2009/125/EG des Europäischen Parlaments und des Rates vom 21. Oktober 2009 zur Schaffung eines Rahmens für die Festlegung von Anforderungen an die umweltgerechte Gestaltung energieverbrauchsrelevanter Produkte (Neufassung), http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2009:285:0010:0035:DE:PDF .

EU (2010a): Verordnung (EG) Nr. 66/2010 des Europäischen Parlaments und des Rates vom 25. November 2009 über das EU-Umweltzeichen, http://eur-lex.europa.eu/legal-content/DE/TXT/PDF/?uri=CELEX:32010R0066&from=EN .

EU (2010b): Richtlinie 2010/30/EU des Europäischen Parlaments und des Rates vom 19. Mai 2010 über die Angabe des Verbrauchs an Energie und anderen Ressourcen durch energieverbrauchsrelevante Produkte mittels einheitlicher Etiketten und Produktinformationen (Neufassung), http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2010:153:0001:0012:DE:PDF .

Manhart (2012): Manhart, A.; Riewe, T.; Brommer, E.; Gröger, J.; PROSA Smartphones - Entwicklung der Vergabekriterien für ein klimaschutzbezogenes Umweltzeichen. Studie im Rahmen des Projektes „Top 100 – Umweltzeichen für klimarelevante Produkte"; gefördert durch: Bundesministerium für Umwelt, Naturschutz und Reaktorsicherheit (BMU), Berlin / Projektträger Jülich; 2012.

RAL (2010): Blauer Engel für umweltschonenden Schiffsbetrieb, RAL-UZ 110, Sankt Augustin, 2010; https://www.blauer-engel.de/sites/default/files/raluz-downloads/vergabegrundlagen_de/UZ-110.zip, Zugriff am 15.01.2015.

Rubik et. al (2005): Rubik, F., Scheer, D., Integrierte Produktpolitik (IPP) in ausgewählten Ländern Europas, Schriftenreihe des IÖW 179/05, Heidelberg, Februar 2005.

2 Green Software Engineering - Modelle und Perspektiven

Autor: Prof. Dr. Stefan Naumann, Institut für Softwaresysteme in Wirtschaft, Umwelt und Verwaltung, Umwelt-Campus Birkenfeld der Hochschule Trier, Postfach 1380, D-55761 Birkenfeld, s.naumann@umwelt-campus.de

2.1 Vorbemerkung

Der Text bezieht sich auf die unter http://oekotop100.de/software/programm/ unter dem Namen des Autors zum Download bereitgestellten Vortragsfolien, welche jeweils mit der entsprechenden Foliennummer referenziert werden.

2.2 Was ist Green Software Engineering?

Im Vortrag wurden aktuelle Trends und Perspektiven des Green Software Engineering vorgestellt. Unter Beachtung des Zitats von Niklas Wirth „Software is getting slower more rapidly than hardware becomes faster“ („A Plea for Lean Software“, Computer 28, 1995) lässt sich die Beobachtung machen, dass zwar mittlerweile der Bereich „Green IT“, also die Entwicklung und Nutzung energieeffizienterer Hardware, im Mainstream von Forschung und Industrie angekommen ist, aber der Bereich der energieeffizienten Software in Forschung und Praxis noch in den Kinderschuhen steckt. So zeigt ein Vergleich der verschiedenen Ressourcenbedarfe des Betriebssystems Microsoft Windows, dass Windows 8 im Vergleich zu Windows 95 die 40fache Prozessorleistung benötigt, den 250fachen Hauptspeicherbedarf hat und 320-mal mehr Festplattenkapazität erwartet (Folie 3). Es stellt sich die Frage, ob der Funktionsumfang in gleichem Umfang gestiegen ist. Des Weiteren lässt sich beobachten, dass bisher zwar Labels und Zertifizierungen – wie es sie seit langem schon für Elektro- und Küchengeräte gibt – für den Bereich der Hardware existieren, im Softwarebereich ein solches Label oder eine solche Kennzeichnung jedoch nicht verfügbar ist. Allerdings gibt es eine Reihe von (häufig selbsterstellten) Labels für Websites, in dem die Betreiber bspw. damit werben, dass die Website grünen Strom auf Serverseite verwendet. Entsprechende Labels werden teilweise auch vom Stromanbieter bereitgestellt. Insgesamt stellt sich beispielsweise die Frage, ob perspektivisch diese Aussagen auch durch Tests hinsichtlich der Energieeffizienz überprüft und veröffentlicht werden können, beispielsweise im Magazin Öko-Test (Fotomontage Folie 6).

Es lässt sich feststellen, dass energiebewusste Software im direkten Nutzen für den Anwender besonders relevant ist für Mobile Systeme (Akkulaufzeit Smartphone etc.), Embedded Systems (Sensoren mit geringer Ressourcenausstattung etc.) sowie für den Bereich des energieintensiven „High Performance Computing“ („Predict the climate change, not produce it“). Aus einer allgemeineren Perspektive sind sämtliche Softwaresysteme relevant, da jede Software in ihrer Ausführung zu einem Energie- und Ressourcenverbrauch führt.

Ausgehend von diesen Überlegungen lassen sich zwei Definitionen benennen, welche die Produkt- und die Prozesssicht widerspiegeln:

(1) Grüne und nachhaltige Software ist Software, deren

- direkte und indirekte negative Auswirkungen
- auf Menschen, Gesellschaft und Umwelt
- über ihren gesamten Lebenszyklus hinweg minimal sind
- und die bestenfalls einen zusätzlichen positiven Beitrag zur nachhaltigen Entwicklung leistet.

(2) Grüne und nachhaltige Softwareentwicklung bedeutet,

- grüne und nachhaltige Software
- mit einem grünen und nachhaltigen Softwareentwicklungsprozess zu entwickeln und

- dabei die positiven und negativen Wirkungen des Softwareprodukts auf die Nachhaltige Entwicklung
- kontinuierlich zu bewerten, zu dokumentieren und für die weitere Optimierung des Softwareprodukts heranzuziehen.

In diesen beiden Definitionen steckt zum einen der Green-IT-Aspekt, also wie sich IKT selber nachhaltiger gestalten lässt, aber auch der Green-by-IT-Aspekt, also die Frage, wie sich mittels des Anwendungsfeldes der Software in anderen Branchen Energie und Ressourcen einsparen lassen.

Hierauf aufbauend wurde seitens unserer Forschungsgruppe das GREENSOFT-Referenzmodell entworfen (Folien 10 und 11). Das GREENSOFT-Modell ist ein konzeptuelles Referenzmodell für „Grüne und Nachhaltige Software“, mit dem das Ziel verfolgt wird, Softwareentwickler, Administratoren, Beschaffer und Softwarenutzer bei der Herstellung, Wartung, Beschaffung, dem Betrieb und der Nutzung von Software im Hinblick auf Zielstellungen der Nachhaltigen Entwicklung zu unterstützen. Der Ordnungsrahmen des Modells beinhaltet ein ganzheitliches Lebenszyklusmodell für Softwareprodukte, Nachhaltigkeitskriterien und Metriken für Softwareprodukte, Vorgehensmodelle für verschiedene Anspruchsgruppen sowie Handlungsempfehlungen und Werkzeuge, die Anspruchsgruppen beim Entwickeln, Beschaffen, Warten und Nutzen von Softwareprodukten in einer mit den Zielen der Nachhaltigen Entwicklung verträglichen Art und Weise unterstützen.

Der Ordnungsrahmen enthält einen Lebenszyklus für Softwareprodukte, der sich, im Gegensatz zu klassischen Softwarelebenszyklen, am Lebenszyklusdenken bzw. Life Cycle Thinking orientiert. Das Lebenszyklusdenken folgt dem Motto „von der Wiege bis zur Bahre“ bzw. „von der Wiege bis zur Wiege“. Es hat zum Ziel, die ökonomische Verträglichkeit, Humanverträglichkeit, Sozialverträglichkeit und die ökologische Verträglichkeit eines Produktes über dessen gesamten Lebensweg hinweg zu bewerten. Der Lebenszyklus beginnt mit den sehr frühen Phasen der Produktentwicklung und endet mit der Entsorgung bzw. dem Recycling. Die Ergebnisse aus den Bewertungen können für eine ausgeglichene Produktoptimierung oder für Vergleiche mit Konkurrenzprodukten herangezogen werden

Der Modellteil „Nachhaltigkeitskriterien und Metriken“ deckt allgemeine Metriken und Kriterien zur Messung der Softwarequalität ab (International Standard ISO/IEC 25000:2005 (E)) und erlaubt die Klassifikation von Kriterien und Metriken für die Bewertung der Wirkungen eines Softwareproduktes auf die Nachhaltige Entwicklung. Passende Kriterien und Metriken umfassen Modelle für die Messung der Softwarequalität im Hinblick auf die Produkt- und Prozessqualität, aber auch Methoden, die der Lebenszyklusanalyse entlehnt sind. Ein wesentliches Unterscheidungsmerkmal sind hier die unmittelbaren Kriterien und Metriken, die sich auf Effekte erster Ordnung bzw. Bereitstellungseffekte beziehen und die mittelbaren Kriterien und Metriken, die sich auf Effekte zweiter und dritter Ordnung bzw. Nutzungseffekte und systemische Effekte beziehen.

Der Modellteil „Vorgehensmodelle“ ermöglicht die Einordnung von Vorgehensmodellen, die die Beschaffung, Entwicklung, Wartung, Administration und Nutzung von Software vor dem Hintergrund der Nachhaltigen Entwicklung unterstützen. Als beispielhafte Ausprägung wurde hier eine generische Erweiterung für Softwareentwicklungsprozesse vorgeschlagen, die auf eine systematische Berücksichtigung von Nachhaltigkeitsaspekten in Softwareprojekten abzielt.

Der letzte Modellteil bildet Handlungsempfehlungen und Werkzeuge ab. Diese sollen die Anspruchspersonen bei der Anwendung von Techniken unterstützen, die nachhaltige Zielstellungen befördern. Dabei sollen die unterschiedlichen Kenntnisstände und Erfahrungswerte der Akteure berücksichtigt werden. Mögliche Rollen in diesem Bereich sind: Softwareentwickler, Beschaffer, Administratoren sowie professionelle und private Anwender, aber auch generell alle an Softwareprodukten beteiligten Akteure.

2.3 Messung des Energieverbrauchs von Software

Ein zentraler Punkt bei der Bewertung der Nachhaltigkeit von Software ist die Frage ihres Ressourcenverbrauchs, und zwar insbesondere in der Nutzungsphase. Die Messung dieses Verbrauchs für ein Artefakt wie Software ist aufgrund ihrer Besonderheiten nicht trivial, da eine Reihe von Abhängigkeiten bestehen. Zunächst ist festzustellen, dass Software einer Reihe von externen Einflüssen ausgesetzt ist:

- Hardware und Stromqualität
- Entwicklungsbedingungen bei der Produktion
- Einbindung in das Smart Grid
- Nutzungsszenarien und Nutzungsverhalten
- Verarbeiteter Content
- Anwendungsbereich der Software (Green by IT)

Aus Sicht der Software-Architektur ist zu beachten:

- Lokal, verteilt, als Service?
- Verwendete Programmiersprache, Compiler und Interpreter
- Laufzeitumgebungen und Betriebssysteme
- Lebenszyklus der Software
- Konfigurationen und Versionen der Software

Hinsichtlich des konkreten Messverfahrens lässt sich feststellen:

- Je verteilter das System, desto ungenauer die Messung
- Eine Messung kann direkt, indirekt (bspw. Ladestrom) durchgeführt oder auch mittels Prognosen abgeschätzt werden
- Metriken sind bisher weitgehend offen (bspw. Useful Work Done per Joule)
- Wie interpretiere und vergleiche ich die Ergebnisse?

Erste Lösungsansätze, um diese Probleme einzugrenzen bzw. um erste Schritte zu machen:

- Konzentration auf Standard-Software mit hohen Installationszahlen (Skaleneffekt, vgl. auch Folien 19 zur Einschätzung der Verbräuche im Vergleich von Entwicklungs- und Nutzungs-Phase)
- Berücksichtigung von mobilen Anwendungen, Embedded Systems und High Performance Computing (hohes Nutzerinteresse)
- Betrachtung von multimedialen Anwendungen, die über Mobilfunk ausgeliefert werden (hohes Einsparpotenzial)
- Betrachtung der gleichen Software in unterschiedlichen Konfigurationen und auch Versionen (gute Vergleichbarkeit)
- Betrachtung unterschiedlicher Software, die ein möglichst ähnliches Aufgabengebiet hat (gute Vergleichbarkeit)
- Bei Whitebox-Betrachtungen sollten vor allem Standard-Bibliotheken wie Container-Klassen, Sortierverfahren etc. untersucht werden (hohe Nutzungszahlen)

Im nächsten Schritt ergeben sich hieraus:

- Standard-Messumgebungen
- Sammlung und Konsolidierung der Messergebnisse unterschiedlicher Forschungsgruppen
- Entwicklung und Validierung von Metriken, um geeignete Nutzungsszenarien aufbauen zu können

- Weiterentwicklung von Prognosemodellen zur Abschätzung von Verbräuchen, wenn keine Mess-Infrastruktur vorhanden ist sowie zur Verbesserung der Messmodelle.

Konkrete Messungen des Energieverbrauchs von Software wurden unter anderem am Umwelt-Campus der Hochschule Trier vorgenommen. Hierzu wurde ein Messaufbau entwickelt, der ein industrielles Strom-Messgerät integriert (Folie 20-22). Hier wird auf dem System Under Test (SUT) das zu bewertende Anwendungsprogramm ausgeführt. Durch Austausch einzelner Hierarchieebenen ist es möglich, eine Anwendung auf verschiedenen Betriebssystemen, Hardware- und Laufzeitumgebungen zu bewerten oder die Anwendung gegen eine andere Konfigurationsvariante oder ein Konkurrenzprodukt auszutauschen. Der Lasttreiber (LT) generiert statistisch reproduzierbare Lasten und wendet sie auf die zu bewertende Anwendung an. Die elektrischen Verbrauchs- und Leistungsdaten, die Leistungsdaten des SUT und die statistischen Daten des LT werden an zentraler Stelle von der Datenerfassung und Auswertung (DEA) verarbeitet.

Sowohl der Messaufbau als auch das prinzipiell angewendete Messverfahren orientiert sich an der ISO/IEC 14756 (International Standard ISO/IEC 14756:1999), einem Standard zur Performanzmessung und Laufzeiteffizienzbewertung von Datenverarbeitungssystemen. Ein wesentlicher Teil des Standards stellt die Generierung statistisch reproduzierbarer Lasten sicher. Die Teilaspekte für die Bewertung der Performanz- und Laufzeiteffizienz sind für die reine Energieeffizienzmessung nicht erforderlich, können aber als zusätzliche Informationsquelle herangezogen werden.

So hat exemplarisch die Messung des Content Management Systems Joomla ergeben, dass bei unterschiedlichen Konfigurationen im Bereich des serverseitigen Cachings mit einer gecachten Konfiguration durchschnittlich 8,6% Energie gegenüber der ungecachten Konfiguration eingespart werden können.

Über die Messungen während der Nutzungsphase hinaus ist es sinnvoll, bereits während der Entwicklung von Software auf den Trend des Energieverbrauchs zu achten. Hierzu bietet sich die Integration der Verbrauchsmessungen in die häufig zur Anwendung zu kommende Continuous Integration an. Der Entwickler bekommt dann bspw. nach dem Nightly Build am nächsten Morgen nicht nur eine Liste von Fehlern bzw. erfolgreichen und gescheiterten Tests, sondern auch eine Übersicht, ob sich hinsichtlich des Energieverbrauchs etwas geändert hat (Folie 26). Insgesamt sind Nachhaltigkeitsaspekte während des gesamten Entwicklungs- und Lebenszyklus der Softwareentwicklung zu berücksichtigen (Folie 27) und können ganz verschiedene Aspekte wie Netzlast-Reduktion oder Verwendung von Energiespar-Pattern umfassen.

2.4 Ausblicke und Forschungsansätze

Um dem Nutzer deutlich zu machen, dass auch Software Energieverbräuche induziert, ist es sinnvoll, diese zu visualisieren. So kann der Green Power Indicator dem Websiten-Besucher anzeigen, ob die angesurfte Website mit regenerativ gewonnenem Strom betrieben wird (bzw. der Provider solchen verwendet). Forscher der TU Dresden haben prototypisch eine Anwendung vorgestellt, welche für Apps ein Kennzeichnungssystem ähnlich dem von weißer Ware vorsieht. In einer Nutzerbefragung des Umwelt-Campus, an der 308 Personen teilgenommen haben, haben sich 56 % für die Labelung von Websites ausgesprochen, und 31% für eine Ampel-Kennzeichnung (Folie 32).

Insgesamt lässt sich feststellen, dass in Forschungslandschaft Bewegung ist, was durch mittlerweile mindestens sieben einschlägige internationale Workshops, ein Sonderheft von IEEE Software zum Thema „Green Software“ sowie zahlreiche weitere Publikationen belegt ist (Folien 33-35). Des Weiteren existiert seit 2013 ein internationaler „Green Code Challenge“, bei dem sich Studierendengruppen untereinander in der Energieoptimierung von Code und Software messen können.

Insgesamt sehen wir drei wesentliche Forschungsstränge, welche die drängendsten Fragen zur Nachhaltigen von Software adressieren:

(1) Was ist energieeffiziente Software?

- Benötigt werden reproduzierbare Maße, Metriken und Nutzungsszenarien
- Benötigt werden energieeffiziente Software-Architekturen
- Welche Rolle spielt Suffizienz im Kontext von Software?

(2) Wie kann energieeffiziente Software produzieren werden?

- Benötigt werden Vorgehensmodelle, die „grüne“ Ideen enthalten, oder sogar neue Vorgehensmodelle

(3) Wie kann energieeffiziente Software befördert werden?

- Beispielhaft sind hier zu nennen: Kundenanforderungen, Entwicklungsumgebungen, Normen, Zertifikate, Lehre, Forschungsmittel

2.5 Weiterführende Informationen

Ausgewählte Veröffentlichungen zum Thema:

Capra, E.; Francalanci, C.; Slaughter, S. A. (2012): Measuring Application Software Energy Efficiency. In: IT Professional, S. 54 - 61.

Steigerwald, B.; Chabukswar, R.; Karthik, K.; Jun, D. V. (2007): Creating Energy-Efficient Software. Hg. v. Intel Corporation. Online verfügbar unter http://software.intel.com/en-us/articles/creating-energy-efficient-software-part-1/, zuletzt aktualisiert am 04.10.2008, zuletzt geprüft am 09.12.2014.

Lami, G.; Fabbrini, F.; Fusani M. (2012): Software Sustainability from a Process-Centric Perspective. In: D. Winkler, R.V. O'Connor und R. Messnarz (Hg.): EuroSPI 2012, CCIS 301: Springer, S. 97–108.

Naumann, S.; Dick, M.; Kern, E.; Johann, T. (2011): The GREENSOFT Model: A Reference Model for Green and Sustainable Software and its Engineering. In: SUSCOM 1 (4), S. 294–304. Online verfügbar unter doi:10.1016/j.suscom.2011.06.004, zuletzt geprüft am 16.11.2011.

Kern, E.; Dick, M.; Naumann, S.; Hiller, T. (2014): Impacts of software and its engineering on the carbon footprint of ICT. In: Environmental Impact Assessment Review, Abstract and excerpt available online: http://dx.doi.org/10.1016/j.eiar.2014.07.003, available since: 19 August 2014, in Press, ISSN 0195-9255.

Drangmeister, J.; Kern E.; Dick, M.; Naumann, S.; Sparmann, G.; Guldner, A. (2013): Greening Software with Continuous Energy Efficiency. In: Horbach; Matthias (Ed.): Informatik 2013. Informatik angepasst an Mensch, Organisation und Umwelt. Lecture Notes in Informatics (LNI) – Proceedings, Volume P-220. Koblenz 2013, S. 940-951.

Weitere Informationen finden sich zudem auf der GREENSOFT-Projekt-Webseite www.green-software-engineering.de

3 Einfluss von Software auf Energie- und Ressourceneffizienz

Autor: Dr. Wolfgang Lohmann, Consultant und freier Forscher für Informatik und Nachhaltigkeit, Fachgruppe Green IT, Schweizer Informatikgesellschaft

3.1 Einführung

Der vorliegende Text stellt die Verschriftlichung des Vortrags „Einfluss von Software auf Energie- und Ressourceneffizienz“ dar, den der Autor auf dem Fachgespräch „Nachhaltige Software“ am 28.11.2014 gehalten hat. Der Vortrag selbst basiert auf einem auf dem Wissenschaftsforum Green IT am 11.12.2013 in Berlin gehaltenen Referat zum Thema „Was ist Grüne Software? Eine Einführung in die Potenziale zur Ressourceneinsparung durch Software“.

Die Kernaussagen lauten:

- Je weiter man bei der Betrachtung von Software von der Hardwareebene abstrahiert, desto relevanter werden Einspar-/ Verlustpotenziale.
- Bei den direkten Effekten durch Software gibt es durchaus Einsparpotenzial. Das Einspar-/ Verlustpotenzial durch indirekte Effekte (vgl. Hilty, 2014, Ausführungen weiter unten) ist bei stark softwarebasierten Systemen jedoch wesentlich größer.
- Bei allen Maßnahmen zur Optimierung sollten die Größenordnungen der Einsparungen ins Verhältnis zum Aufwand gesetzt werden. Für direkte Effekte ist der Strommix zu beachten.

3.2 Beschreibung von Softwaresystemen

Software umfasst, nach dem Verständnis des Autors, sämtliche Arten der Beschreibung von Information und der Beschreibung der Bearbeitung von Information. Software wird gerne als virtuell bezeichnet. Sie ist zwar nicht an ein spezielles Material gebunden, aber sie benötigt immer Material oder physikalische Größen als Träger, sei es für die Darstellung von Texten oder Diagrammen, die Speicherung auf Datenträgern oder Halbleiterspeichern oder den Transport durch Netzwerke.

Je nach Art der Anwendung wird Software in IT-Systemen oft in Schichten eingeteilt, beispielsweise in die Schichten Betriebssystem, Middleware (z. B. Software Plattformen wie Java oder .NET), Anwendungssoftware (lokal, z. B. Textverarbeitung), verteilte Anwendungssoftware über Rechnergrenzen hinaus, die abstrakt und virtuell über der Hardware zu schweben scheinen. Und dennoch bewirkt Software Ressourcenverbrauch (insbesondere Energieverbrauch als die am meisten offensichtliche Ressource) und Materialflüsse.

Unabhängig von der jeweils adressierten Schicht ist die Beschreibung von Softwaresystemen ein komplexer Vorgang, den man mit Abstraktionen zu beherrschen versucht, die dem Denken der Menschen für die jeweilige Problemdomäne näher sind als die zugrundeliegende Rechnerarchitektur. Diese Abstraktionen werden schrittweise verfeinert, um eine operative Ausführung zu ermöglichen. In den obersten Abstraktionsschichten erfolgt dies meist durch Menschen, die das zu beschreibende System in Module zerlegen. Die Inhalte der Module können mithilfe von Entwurfsmustern beschrieben werden, die wiederum beispielsweise objektorientierte Sprachkonzepte nutzen. Schrittweise wird das Problem auf die Ebene von Maschinenbefehlen überführt, i. d. R. durch Programme, die man Übersetzer/Compiler nennt. Das Ergebnis ist eine lineare Folge von Instruktionen, deren Reihenfolge aber nicht die Folge der Abarbeitung durch Hardwarekomponenten, z. B. eine CPU (central processing unit), widerspiegelt.

3.3 Von der Software zum Energieverbrauch durch Hardware

Auf der Ebene der Hardware erscheint Software als eine Folge von logischen Bitwerten, die durch physikalische Größen wie beispielsweise höhere (high) oder niedrigere (low) elektrische Spannungen

dargestellt werden. Die Reihenfolge dieser physikalischen Werte bestimmt Zustandsänderungen der Hardware, welche in der Summe zu dem gewünschten Verhalten eines Systems führen. Sowohl die Zustandsänderungen der Hardware als auch das statische Speichern von logischen Bitwerten sind mit einem Energieverbrauch verbunden.

Der Energieverbrauch von integrierten Schaltkreisen kommt zu einem Großteil von logischen Gattern, die aus Transistoren und Kondensatoren bestehen. Im Beispiel von CMOS-Schaltungen (CMOS – complementary metal-oxide-semiconductor) entstehen hier Energieverluste beim Laden/ Entladen der Kondensatoren, Verluste durch Kurzschlussströme während des Schaltvorgangs und Leckverluste (engl. leakage current) durch das (nie vollkommene) Sperrverhalten von Transistoren (genauere Beschreibungen finden sich in der Literatur z. B. KYUNG u. a., 2011; Rabaey u. a., 2003; Narendra u. a., 2006; Padeffke, 2005, Anhang A).

Der Energieverbrauch eines Prozessors hängt stark von den Instruktionen ab, die er ausführt. Der Energieverbrauch einer Instruktion selber hängt wiederum von den konkreten Sequenzen ab (Instruktionscache), der Datenverfügbarkeit, den Operation auf Bitmustern, den innerhalb des Prozessors in Anspruch genommenen Logikbausteinen etc.

3.4 Direkter durch Software verursachter Energieverbrauch

Der Energieverbrauch entsteht durch elektrische Leistungsaufnahme über eine Zeitdauer. In einer Vielzahl von Fällen führt daher eine Optimierung der Algorithmen, mit dem Ziel, die Rechenzeit zu verkürzen, zu Energieeinsparung. Viele Beispiele zeigen allerdings auch, dass Energieeinsparungen durch Algorithmen, die die Leistungsaufnahme reduzieren, möglich sind, obwohl sich die Laufzeit verlängert.

Eine direkte Implementierung eines Algorithmus in Hardware ist am effizientesten[2]. So kann beispielsweise eine MP3-Codierung[3] und Decodierung sehr energieeffizient auf Chipebene erfolgen. Wegen der fehlenden Flexibilität für generelle Aufgaben kommt dies i. A. nicht in Betracht und hätte auch noch andere gravierende Nachteile, z. B. die schnellere Veraltung der Hardware.

Bei der Nutzung eines Programms konzentriert sich die Abarbeitung von Befehlen oft auf einen Teil der Funktionalität und entsprechend auf beschränkte Bereiche im Code. Beispielsweise ist ein Programm zur Textverarbeitung im Wesentlichen damit beschäftigt, auf die Tastatureingabe zu warten und den eingegebenen Text auf dem Bildschirm darzustellen. Indem man sich auf solche Hotspots konzentriert, erzielt man die größten Effizienzeffekte. Selten durchlaufende Programmteile sind dagegen vernachlässigbar.

Der Weg hin zu mehr Abstraktion ermöglicht die Beschreibung immer komplexerer Probleme. Gleichzeitig verringern sich die Abhängigkeiten von der konkreten Hardware. Diese steigende Abstraktion hat jedoch Nachteile hinsichtlich des Energieverbrauchs (und der Geschwindigkeit), wie z. B. in Einreichungen für Workshops der Reihe GREENS (z. B. im Jahr 2012) (Götz u. a. 2012) dargestellt wurde. Dies liegt beispielsweise daran, dass mit jeder höheren Abstraktionsstufe der organisatorische Aufwand wächst, den ein Compiler in die übersetzten Befehlssequenzen einbauen muss, um beispielsweise das objektorientierte Verhalten wie Vererbung zu simulieren. Durch die Unabhängigkeit von der Hardware bei steigender Abstraktion verringert sich auch das Wissen, mit dem energieeffizientere

---

[2] Zusätzlich bietet Hardware noch viel Effizienzpotenzial: Feynman (Feynman u. a., 2000) berechnete, dass mit 1 Watt theoretisch 10^{18} Bit-Operationen pro Sekunde möglich seien. Das entspräche der Leistung von 1 000 000 Pentiumprozessoren bei etwa einem Energieverbrauch eines Nokia 3210 von 2000 (Tsiao, 2012, Slide w/ notes 2).

[3] MP3 = Audio Komprimierungsverfahren MPEG-1/2 Audio Layer III benannt nach der Moving Picture Experts Group (MPEG)

Übersetzungen generiert werden können. Gleichzeitig wächst aber das Wissen über die Programm- und Problemstruktur und ihren Kontext. Eine sinnvolle Entscheidung auf Architekturebene kann daher wesentlich mehr Energie sparen. Ein dynamisches Verschieben von Prozessen und eine damit herbeigeführte Konsolidierung auf weniger Server mit der Folge, temporär ungenutzte Server abzuschalten, bietet wesentlich größeres Einsparpotenzial als eine Optimierung auf Quellcodeebene.

Weitere Aspekte, die in diesem Zusammenhang eine Rolle spielen, sind die Einflüsse von Übersetzungsprozessen. Gleiche Compiler erzeugen für gleiche Programme unterschiedliche Code-Sequenzen für verschiedene Hardware, verschiedene Compiler generieren verschiedenen Code für gleiche Programme auf gleicher Hardware. Algorithmen haben auf verschiedene Hardware verschiedene Effekte, die sich je nach Daten unterscheiden. Programme, die durch Interpreter ausgeführt werden, führen zu einer erhöhten Energieaufnahme, bieten aber mehr Möglichkeiten, dynamisch auf aktuelle Kontexte zu reagieren. Die Schichten der Software sind stark miteinander verwoben. Während der Ausführung kommt es hier zu komplexen Wechselwirkungen. Allgemeine Lösungen im Übersetzungsprozess zur Optimierung der Energieeffizienz des auszuführenden Programms sind daher schwer zu erreichen.

3.5 Umweltauswirkungen durch erforderliche Hardware

Bei der Betrachtung des Einflusses von Software auf die Umwelt liegt der Fokus oft auf dem (naheliegenden) direkten Energieverbrauch der Hardware während der Nutzungsphase. Häufig wird vergessen, dass Software das Vorhandensein oder die Erneuerung von Hardware erfordert und demzufolge auch mit verantwortlich ist für die daraus folgenden Umweltauswirkungen.

Zunächst ist auch für die Produktion und die Entsorgung von Hardware Energie erforderlich. Die Produktion erfordert zudem Rohstoffe. Für die Herstellung eines Smartphones werden derzeit beispielsweise 55 Elemente eingesetzt (Wäger u. a., 2010). Die Gewinnung solcher Rohstoffe geht oft mit Umweltschäden und schlechten Arbeitsbedingungen einher. Ebenfalls dadurch bedingte Konflikte haben zur Prägung des Begriffes „Konfliktmineralien“ geführt. Die Entsorgung von IT (Informationstechnik) erfolgt oft in Ländern Afrikas oder in Indien. Recycling wird dort unter gesundheitsgefährdenden Bedingungen und ohne Rücksicht auf die Umweltauswirkungen durchgeführt (z. B. Freisetzen von Schwermetallen wie Blei, Kadmium und Quecksilber). Die Ausbeute beschränkt sich meist auf gängige Rohstoffe wie Kupfer oder Gold. Diese Prozesse laufen in Europa geordneter, aber auch hier sind die Recyclingquoten von seltenen Erden sehr gering.

Die Auswirkung des Energieverbrauches der Hardware wurde in einer Studie (Prakash u. a., 2012) am Beispiel der Amortisation eines Notebooks unter energetischen Gesichtspunkten untersucht (zwischen 7- 88 Jahren, je nach verwendeten Daten und untersuchtem Szenario). Hischier und Kollegen (Hischier u. a., 2014) verglichen die Umweltauswirkungen eines Desktop-Rechners, eines Laptops und eines Tablets, mit dem Ergebnis, dass die Nutzungsphase einen immer kleineren Anteil an der Gesamtenergie einnimmt. Dies gilt insbesondere, wenn die Nutzung von Dienstleistungen aus dem Internet hinzugenommen wird. Ähnliche Zahlen lassen sich auch bei Ericsson (Malmodin u. a., 2014) finden. Der direkte Anteil der Software steckt im Energieverbrauch in der Nutzungsphase. Folglich ist der Effekt lokaler Optimierungen des Energieverbrauchs mittels Programmierung beschränkt. Für eine Betrachtung des Einflusses von Software auf die Umwelt, insbesondere auf das Klima, ist auch der jeweilige Strommix zu beachten.

3.6 Nachhaltigkeit durch IKT

Die Beobachtungen zum Energie- und Ressourcenverbrauch durch Hard- und Software passen gut in das Rahmenwerk zur Nachhaltigkeit durch die Informations- und Kommunikationstechnologie (IKT) (Hilty u. a., 2014). Die IKT ist selbst Teil des Problems, weil durch sie Umweltprobleme verursacht

werden. Ebenso kann das Problem mithilfe der IKT angegangen werden. Die Auswirkungen werden dabei nach ihrer Wirkung unterschieden und wie folgt eingeteilt:

1. Direkte Effekte (Effekte erster Ordnung),
2. Effekte, die andere Auswirkungen ermöglichen (engl. enabling effects, auch Effekte zweiter Ordnung) und
3. systemische Effekte (Effekte dritter Ordnung).

Direkte Effekte bezeichnen Auswirkungen, die durch die Produktion, während der Nutzungsphase oder durch die Entsorgung von Technik entstehen und mittels Lebenszyklusanalyse bestimmt werden können. Die Effekte zweiter Ordnung bezeichnen Auswirkungen, die durch die Anwendung der IKT entstehen, also Induktions-, Obsoleszenz-, Substitutions- und Optimierungseffekte. Systemische Effekte bezeichnen Langzeitreaktionen der dynamischen sozio-ökonomischen Systeme auf die Verfügbarkeit von IKT-Dienstleistungen, beispielsweise neue Formen des Konsumverhaltens.

Auch im Zusammenhang mit Software sind direkte Effekte zwar relevant, Effekte zweiter und dritter Ordnung übersteigen diese aber in der Wirkung um Größenordnungen. Anwendungen, die eine positive Auswirkung auf die Umwelt haben, kann man der Überschrift *Grün durch Software* zuordnen. In Bezug auf Software ist die Verhinderung von Hardware-Obsoleszenz von erheblichem Interesse, da es aufgrund des hohen Herstellungsaufwandes (siehe oben) besonders wünschenswert ist, Hardware möglichst lange zu nutzen. Software (-innovation) ist einer der größten Verursacher von Hardwareverschleiß, indem eigentlich weiterhin funktionstüchtige Geräte aufgrund geänderter Anforderungen und Wünsche außer Betrieb genommen werden. Die Wechselwirkungen von Software mit neuen Hardware-Fähigkeiten sind allerdings oft schwer voneinander zu trennen. Wie man Softwareentwicklung dazu bewegen kann, Hardware-Obsoleszenz zu reduzieren, ohne die Innovation der Software zu verhindern, ist nicht klar und sollte näher erforscht werden[4].

Ein großes Potenzial bietet die Anwendung von Software zur Einsparung/ Optimierung in anderen Bereichen, wie im SMARTer2020 Report deutlich beschrieben wurde (GeSI, 2012). Optimierungen von Industrieprozessen, beispielsweise durch Simulationen, das Bestimmen optimaler Transportrouten und viele Anwendungsmöglichkeiten bieten enormen Raum zur Ressourceneinsparung. Substitutionseffekte sind besonders als Beitrag zur Dematerialisierung interessant, wie im Beispiel der als MP3 über das Internet übertragbaren Musik, und damit der Einsparungen von CD-Produktion, Material sowie Transportkosten. Die Virtualisierung ist ebenfalls ein solches Beispiel, indem Hardware ersetzt wird oder gar nicht erst angeschafft werden muss.

Systemische Effekte wie Reboundeffekte, aber auch das Entstehen von anderem Konsumverhalten, sind am schwersten vorhersehbar, haben aber die größten Einflüsse. Nachhaltige Software muss diese systemischen Effekte im Blick haben und darf nicht zu einer Verstärkung des Energie- und Ressourcenverbrauches führen.

3.7 Immaterieller Charakter der Software

Software ist in ihrem Wesen ganz anders als die meisten gegenständlichen Produkte. Sie ist ein komplexes System von voneinander abhängigen Informationsbausteinen. Als passendere Bezeichnung schlägt der Autor daher „Brittleware“ vor: Änderungen am Code, am Quellcode, in Datenformaten, in der Struktur oder Kontext führen schnell zum „Zerbrechen von Software“. Die Komplexität, die

[4] Mögliches Kriterium: "nachhaltige" Software läuft zum Zeitpunkt der Erstauslieferung auch auf Hardware, die zu diesem Zeitpunkt schon 3 Jahre alt ist.

Wechselwirkungen von gleichzeitig ablaufenden Prozessen machen es schwer, Aussagen zur Optimierung zu finden. Dennoch ist Software auch weich, weil sie aktualisierbar ist.

Software ist eine virtualisierte Regeleinheit eines Systems. Die Steuerung von Geräten kann an neue Aufgaben angepasst werden. Nur wenige Produkte können vergleichbar nach Auslieferung verbessert werden, z. B. bei Feststellen eines Fehlers im Produkt. Dies vermeidet beispielsweise auch das Veralten von Hardware. Neue Funktionalität kann durch Software nachgereicht werden.

Beispielsweise kann ein Programm zur Optimierung der Energiemanagementeinstellungen eines Laptops/ Desktop-Rechners nach der Veröffentlichung sofort weltweit genutzt werden. Dadurch trägt es weltweit dazu bei, Energie zu sparen. Ein ebenfalls starker Skalierungseffekt ist möglich, wenn Verbesserungen bei der Übersetzung von Softwarebeschreibungen gefunden werden. Könnte man beispielsweise die Übersetzung von Entwurfsmustern energetisch optimieren, dann steht diese Verbesserung jedem Programm der unterstützten Programmiersprache offen. Durch die Skalierbarkeit gewinnen auch kleine Optimierungsgewinne durch ihre Summe an Bedeutung.

Eine Software ist ein Wissens-Produkt. Analog zum Wissen können Menschen weltweit gleichzeitig an Softwareverbesserungen arbeiten. Daraus resultiert auch die dramatische Skalierbarkeit. Hier liegen besonders Chancen von Freier Software und von Open Source Software.

3.8 Schlussfolgerungen

Der größte Einfluss von Software ist systemischer Natur, d. h. Änderungen des Verhaltens von Menschen, neuartige Prozesse und Produkte wie Online-Handel. *Grün durch Software* könnte die größten Einsparungen erbringen. Es ist dennoch immer sinnvoll, Energie- und Ressourcen zu sparen, wenn dies mit akzeptablem Aufwand möglich ist. Um größere Einsparungen zu erreichen, sollte eine Priorisierung vorgenommen werden. Dabei gilt: Je höher die Abstraktionsstufe, desto höher sind voraussichtlich die erwarteten Einsparpotenziale.

Folgende weitere Schritte hin zu einer nachhaltigen Softwareentwicklung werden vorgeschlagen:

- Für eine Diskussion zu nachhaltiger Software müssen klarere Zielvorgaben entwickelt werden: Was möchte man in diesem Fall unter diesem Begriff verstehen? Bezieht er sich auch auf *Grün durch* Software? Was sind die konkreten Einsparziele? Wo liegen die Betrachtungsgrenzen? Wie ist der jeweilige Zeithorizont?
- Nachhaltigkeit als Attribut der Software in den Anforderungen der Softwareentwicklung: Hier besteht Forschungsbedarf, wie diese Nachhaltigkeitsziele definiert werden können.
- Awareness für Softwarearchitekten: Je abstrakter die Ebene ist, auf der die Software erstellt wird, desto stärkere Auswirkungen kann dies auf den Energieverbrauch haben, z. B. Vermeidung unnötiger Datenabfragen im Internet oder Verteilung von Aufgaben auf verschiedene Rechenzentren je nach Situation. Wissen über Nachhaltigkeit ist für Softwareentwicklung auf dieser Ebene notwendig. Im Zweifel sollte der Schwerpunkt auf die Erforschung von Methoden gelegt werden, mit denen man nachhaltige Systeme entwirft, d. h. man sollte eher auf der abstrakteren Entwurfsebene ansetzen als auf Quellcodeebene.
- Forschung an Übersetzungswerkzeugen ist notwendig: Die Überbrückung der Abstraktionsstufen sollte Aufgabe von automatischen Werkzeugen bleiben. Energiebewusster Code ist Aufgabe der Grundlagenforschung an Informatikwerkzeugen, d. h. Forschung an Übersetzungsprozessen, die aus Problembeschreibungen möglichst energieeffizienten Code generieren, möglicherweise adaptiv und energiebewusst (energy aware). Sprachen und Werkzeuge, die den Aspekt Energie berücksichtigen, sollten entsprechend einfach zu verwendende Abstraktionen bereitstellen. Skalierungsfähigkeit müsste mehr ausgenutzt werden - ein Werkzeug ist sofort weltweit auf alle ähnlichen Probleme anwendbar.
- Der Strommix bestimmt die Relevanz von Einsparungen des direkten Energieverbrauchs.

- Wege zur Identifikation von Energie-Hotspots müssen aufgezeigt werden, an denen eine Optimierung Sinn machen würde, am besten durch automatische Werkzeuge. Dazu werden Metriken und Methoden benötigt.
- Wie vermeidet man Hardware-Obsoleszenz, ohne die Innovation zu stark einzuengen?

3.9 Quellenverzeichnis

Feynman, R. P.; Hey, A. (2000): Feynman Lectures on Computation, ISBN-13 978-0738202969.

GeSI (2012): GeSI SMARTer 2020: The Role of ICT in Driving a Sustainable Future, http://gesi.org/SMARTer2020. Abgerufen am 18.12.2014.

Götz, S.; Wilke, C.; Richly, S.; Aßmann, U. (2012): Approximating Quality Contracts for Energy Auto-Tuning Software, Proc. GREENS 2012, Zurich.

Hilty, L. M.; Aebischer, B. (2014): ICT for Sustainability: An Emerging Research Field. Hilty, L.M., Aebischer, B. (eds.) ICT Innovations for Sustainability. Advances in Intelligent Systems and Computing 310. Springer International Publishing, in press.

Hischier, R.; Coroama, V. C.; Schien, D.; Ahmadi Achachlouei, M. (2014): Grey Energy and Environmental Impacts of ICT Hardware. In: Hilty, L.M., Aebischer, B. (eds.) ICT Innovations for Sustainability. Advances in Intelligent Systems and Computing 310. Springer International Publishing, in press.

Kyung, C.-M.; Yoo, S. (2011): Energy-Aware System Design: Algorithms and Architectures, Springer.

Malmodin, J.; Lundén, D.; Moberg, Å.; Andersson, G. ;Nilsson, M. (2014): Life Cycle Assessment of ICT. Journal of Industrial Ecology, 18: 829–845. doi:10.1111/jiec.12145.

Narendra, S. G.; Chandrakasan, A. P. (Eds.) (2006): Leakage in Nanometer CMOS Technologies. Springer, ISBN 978-0-387-28133-9.

Padeffke, M. (2005): Entwurfsverfahren für asynchrone Schaltungen unter Verwendung von Standardsoftware, Cuvillier, Göttingen, ISBN 3-86537-371-9.

Prakash, S.; Liu, R., Schischke, K.; Stobbe, L. (2012): Zeitlich optimierter Ersatz eines Notebooks unter ökologischen Gesichtspunkten, 2012, Umweltforschungsplan des Bundesministeriums für Umwelt, Naturschutz und Reaktorsicherheit, Forschungskennzahl 363 01 322, UBA-FB 001666, S48

Rabaey, Jan M.; Chandrakasan, A.; Nikolic, B. (2003): Digital Integrated Circuits. Prentice Hall.

Tsao, Shiao-Li (2012): Energy aware Computing, lecture notes 2012, http://brass.cs.nctu.edu.tw/EAC-2012/lecturenotes.html, aufgerufen am 4.01.2015

Wäger, P.; Lang, D.; Bleischwitz, R.; Hageluken, Ch.; Meissner, S.; Reller, A.; Wittmer, D. (2010): Seltene Metalle. Rohstoffe für Zukunftstechnologien. SATW, Schweizerische Akademie der Technischen Wissenschaften.

4 Software im Kontext der Nachhaltigkeit

Autor: Erik Albers, Free Software Foundation Europe e.V., Schönhauser Allee 6/7, 10119 Berlin, eal@fsfe.org

4.1 Zusammenfassung

Dieser Artikel ist im Zusammenhang des Fachgesprächs „Nachhaltige Software“ entstanden, veranstaltet am 28.11.2014 durch das Öko-Institut e.V. sowie das Umweltbundesamt. Er ist ein Beitrag zu der Forschungsfrage „Was ist Nachhaltige Software?“ beziehungsweise zu der Frage, welche Kriterien für die Nachhaltigkeit von Software und deren Entwicklung in Betracht gezogen werden können.

Explizit widmet sich dieser Artikel den nachhaltigen Aspekten Freier Software und offener Entwicklungsmodelle. Dazu wird zuerst in das Verständnis von Nachhaltigkeit und in die Nachhaltigkeit digitaler Ressourcen eingeführt. Es folgt eine Erklärung Freier Software und schließlich wird skizziert, welche direkten positiven ökologischen Auswirkungen durch die Entwicklung und Verwendung Freier Software erzielt werden können.

4.2 Zum Begriff „Nachhaltigkeit“

Je mehr unsere alltäglichen Interaktionen durch die Verwendung digital-technischer Geräte geprägt werden, desto mehr rückt die Ökobilanz dieser technischen Geräte in den Fokus der Aufmerksamkeit. Elektroschrott, Recycling, „Green-IT“, Ökostrom, Halbwertszeit von Hardware und verwandte Aspekte gewinnen an Bedeutung. Über die „Nachhaltigkeit“ der verwendeten Software wird hingegen kaum gesprochen. Was könnte man darunter verstehen, was ist nachhaltig und welche Anforderungen müsste Software erfüllen um nachhaltig zu sein?

Der Begriff „Nachhaltigkeit“ hat im letzten Jahrzehnt eine nahezu inflationäre Verwendung gefunden[5], insbesondere in Politik und Wirtschaft. Als eine Folge daraus wurde in unterschiedlichsten Kontexten von Nachhaltigkeit gesprochen, so dass der Begriff weiter an Unschärfe gewonnen hat. Deshalb wird den weiteren Ausführungen dieses Artikels folgende Definition zu Grunde gelegt:

> *„Dauerhafte Entwicklung ist Entwicklung, die die Bedürfnisse der Gegenwart befriedigt, ohne zu riskieren, dass künftige Generationen ihre eigenen Bedürfnisse nicht befriedigen können.“*

Dieses Zitat ist einer der zentralen und meist zitierten Leitsätze des Brundtland-Berichts *„Unsere gemeinsame Zukunft“* (Brundtland 1987, Absatz 49)[6]. Durch die darin enthaltene intertemporale Dimension wird in Kurzform auch von „Generationengerechtigkeit“ gesprochen.

Bei Betrachtung endlicher Ressourcen, die eines Tages erschöpft sein könnten, wäre unter der Generationengerechtigkeit ein Konsum zu verstehen, der die vorhandenen Ressourcen auf eine Art und Weise schont, dass auch zukünftige Generationen noch von der gleichen Ressource zehren können. Bei Software gilt es, diese Aspekte der Ressourcenschonung beim Energieverbrauch der Software zu bedenken, sowie im Rohstoffverbrauch bei der für den Betrieb der Software notwendigen Hardware. Betrachten wir hingegen Software selbst als Ressource, ergeben sich andere Bedingungen der Generationengerechtigkeit. Denn Software ist digital und damit immateriell – und für den Erhalt digitaler Ressourcen gelten andere Kriterien einer nachhaltigen Entwicklung.

---

[5] Der Autor empfiehlt dazu eine satirische Visualisierung der zunehmenden Verwendung des Begriffes “Sustainable” durch den Webcomic “xkcd”: *http://xkcd.com/1007/*

[6] Der sogenannte *“Brundtland-Bericht”* ist der Abschlussbericht der von den Vereinten Nationen eingesetzten Brundtland-Kommission. Seine Veröffentlichung hat maßgebend zu einem gemeinsamen globalen Verständnis einer nachhaltigen Entwicklung beigetragen.

4.3 Nachhaltigkeit digitaler Ressourcen

In der Digitalen Gesellschaft werden unser Arbeitsalltag, unsere Kommunikation und ein großer Teil unserer Infrastruktur inzwischen von Software gesteuert oder vereinfacht. Heute in Deutschland Heranwachsende ohne Zugang zu jeglichem Computer aufzuziehen wäre geradezu verantwortungslos, da es einem sozialen und gesellschaftlichen Ausschluss gleichkäme. Je mehr kritische Infrastruktur von Software gestellt wird, desto wichtiger werden auch der Erhalt, das Verständnis und der Zugang zu dieser Software. Software wird zu einem Fundament, dessen eigene Existenz zum Erhalt unserer digital gewachsenen Infrastruktur notwendig ist; ebenso wie das Wissen um die Bedienung der Software. Es ist an der Zeit, Software als Informations- und Wissensressource zu verstehen.

Betrachten wir Software als eine Ressource, unterscheidet sich diese grundlegend von natürlichen, endlichen Ressourcen. Software – sowie digitale Information jeglicher Art – ist eine immaterielle Ressource und unterliegt damit ähnlichen Bedingungen wie Wissens- und Kulturgüter. Für digitale Ressourcen gilt, dass trotz exzessiver Anwendung, Gebrauch und miteinander Teilen kein Verzehr der Güter stattfindet. Jeder kann von derselben Ressource profitieren, ohne dabei die Ressource für andere Konsumenten zu schmälern.

Beispiel: Betrachten wir ein einzelnes Computerprogramm als eine Ressourcen-Einheit der Menge Eins, dann kann diese Einheit prinzipiell unendlich oft verwendet, kopiert und getauscht werden, ohne dass sich dadurch die Ursprungsressource jemals verbrauchen würde. Im Gegenteil: Mit jeder Kopie des Computerprogramms steigt die Menge der zur Verfügung stehenden Ressourcen-Einheiten. Betrachtet man eine spezifische Form digitaler Information – zum Beispiel ein einzelnes Computerprogramm – als besonders schützenswert und möchte diese Ressource für zukünftige Generationen erhalten, dann ist eine erfolgversprechende Strategie, möglichst viele Kopien davon (= Einheiten) möglichst weit zu verbreiten.

Zusammenfassend gilt, dass Software dann zu einer nachhaltigen Ressource wird, wenn sie frei kopiert und angepasst werden kann. Diese Möglichkeiten müssen jedoch explizit eingeräumt werden. Denn obwohl Software sich durch die Erstellung von Kopien nicht erschöpfen kann, können digitale Ressourcen durchaus ausschließbare Güter sein: Durch Anwendung von Immaterialgüterrechten, geschlossenem Code und Kopierschutzmaßnahmen werden digitale Ressourcen künstlich verknappt und damit vom Teilen, dem freien Gebrauch und der freien Weiterentwicklung ausgeschlossen. Die künstliche Verknappung des Angebots soll markterschaffend wirken; allerdings liegt gerade in der künstlichen Verknappung digitaler Ressourcen die Gefahr des unwiderruflichen Ressourcen- und damit Wissensverlustes. Denn mit der Nicht-Veröffentlichung des dem Programm zu Grunde liegenden Codes wird das Wissen um die Herstellung und Funktionsweise der Software privatisiert. Das sorgt für einen unmündigen Kunden, der vom Support und dem Gutdünken des Herstellers abhängig ist. In unserem Zusammenhang ist es darüber hinaus vor allem eine Gefahr für den Erhalt der Software als zukünftige Ressource.

Durch das Verschließen des Codes liegt das Wissen um die Funktion sowie das Recht der Kopie und der Verbreitung dieser Information in den Händen einer geschlossenen Gruppe, beispielsweise eines Unternehmens. Verschwindet dieses Unternehmen eines Tages vom Markt, dann ist die Gefahr groß, dass damit auch für immer das Wissen um die Software verloren geht. Gleiches gilt für verschlossene Dateiformate. Daten, die in proprietären Formaten gespeichert werden, können nicht als zukunftssicher gelten, denn es sind nur ganz spezielle, verschlossene Computerprogramme fähig, diese Daten auszulesen. Mehr noch: Proprietäre Dateiformate erschaffen und gestalten Insellösungen und Inkompatibilität. Dadurch sollen Kunden an den Hersteller gebunden werden. Geht jedoch dem Hersteller das Wissen um seine proprietären Formate verloren, ist es damit zugleich den zukünftigen Generationen verloren gegangen. Das ist eine der wenigen Gefahren, wie sich eine digitale Ressource in der Tat erschöpfen kann.

Dieser Gefahr entgegen wirken Freie Software und Offene Standards[7]. Beide stehen der Öffentlichkeit dank ihrer offenen Lizenzbestimmungen frei zur Verfügung. Durch öffentliche Dokumentation sowie das explizite Recht, die Software zu kopieren und anzupassen, bilden sie das Fundament, um digitale Information und Software als eine nachhaltige Ressource für zukünftige Generationen zu erhalten. So lange es Computer gibt, werden diese in der Lage sein, Freie Software zu verstehen und zu verwenden. Das erfüllt nicht nur den Anspruch der Generationengerechtigkeit, sondern auch einen weiteren Leitsatz des Brundtland-Berichts:

> *„Im Wesentlichen ist dauerhafte Entwicklung ein Wandlungsprozess, in dem die Nutzung von Ressourcen, das Ziel von Investitionen, die Richtung technologischer Entwicklung und institutioneller Wandel miteinander harmonieren und das derzeitige und künftige Potential vergrößern, menschliche Bedürfnisse und Wünsche zu erfüllen."*[8]

Verfolgen wir also eine Entwicklung, welche *„die Nutzung von Ressourcen"* auf eine Weise ermöglicht, *„das derzeitige und künftige Potential [zu] vergrößern"*, dann müssen wir auf Freie Software und Offene Standards setzen. Nur diese können die Ressource Software nachhaltig sichern und deren Potenzial auch für die Zukunft vergrößern. Es ist an der Zeit, für die Digitale Gesellschaft eine *Digitale Nachhaltigkeit*[9] zu fördern.

4.4 Nachhaltigkeit 'durch', aber auch 'von' Software

Die bisherigen Ausführungen legen nahe, dass bei der Bestimmung von „nachhaltiger Software" mindestens zwei Aspekte von Nachhaltigkeit berücksichtigt werden sollten. Zum einen die ökologischen Auswirkungen durch den Betrieb von Software, eine Nachhaltigkeit *'durch'* Software. Zum andern das Verständnis von Software selbst als eine Ressource, das eine Nachhaltigkeit *'von'* Software fordert.

Bei der Betrachtung von Nachhaltigkeit *durch* Software ist in erster Linie der Energieverbrauch ein maßgebender Faktor. Um Prozesse durchzuführen, benötigt Software Energie. Energieerzeugung bringt jedoch häufig massive Umwelt- und Klimaeinflüsse mit sich. Eine logische Ableitung aus diesem Zusammenhang wäre, dass nachhaltige Software eine Software sei, die möglichst effizient und dadurch Energie schonend läuft. Betrachten wir als Beispiel Software A und B, welche exakt dieselbe Leistung erbringen, wobei A jedoch weniger Energie für das gleiche Ergebnis benötigt. Dann wäre A in unserem Sinne als eine Energie schonende und damit nachhaltige Software zu verstehen.

Der Energieverbrauch zum Betrieb der Software ist jedoch nur ein Aspekt. Einen weitaus größeren ökologischen Fußabdruck hinterlässt die zum Betrieb von Software produzierte Hardware. Ihre Produktion erfordert hohe Mengen an Energie und oft den Einsatz äußerst begrenzter Ressourcen, z. B. so genannter „seltener Erden". Software, die zum Betrieb möglichst geringe Anforderungen an Hardware stellt oder gar eine langfristige Verwendung von Hardware garantiert, ist damit im Sinne der Ressourcenschonung definitiv auch als eine nachhaltige Software zu verstehen.

Schließlich müssen Software und digitale Information selbst als Ressource betrachtet werden. Software wird immer mehr zur kritischen Infrastruktur der Digitalen Gesellschaft. Die größte Gefahr für

---

7 Unter dem Begriff „Offene Standards" werden Dateiformate oder Protokolle verstanden, deren Dokumentation, Verwendung und Implementation frei, offen und für alle zugänglich ist. Für eine genaue Definition siehe die „Genfer Erklärung zu Standards und der Zukunft des Internets" (Genf 2008)

8 Brundtland 1987, Absatz 15

9 "Digitale Nachhaltigkeit" ist ein noch junger Begriff ohne allgemeingültige Definition. In der Wissenschaft (vgl. Busch 2008, Grassmuck 2004, Stürmer 2009) hat sich jedoch eine weitgehend einheitliche Verwendung des Begriffes etabliert (vgl. Martens 2013). Demnach wird unter Digitaler Nachhaltigkeit mindestens verstanden: die Verwendung Freier Software und Offener Standards, der offene Zugang zu Daten und die Freie Zirkulation von Daten.

die Nachhaltigkeit *von* Software besteht – analog zu der Ressource Wissen – darin, die eigentlich unbegrenzte Ressource in einer Weise zu verknappen oder zu verschließen, dass zukünftige Generationen nicht mehr daran teilhaben können. Im Gegenzug können sich insbesondere in der Verwendung von Software als eine gemeinsame Ressource erhebliche Synergieeffekte ergeben, da effiziente Lösungen geteilt und damit potenziert werden[10]. Wie zu sehen sein wird, kann Freie Software zu allen drei Aspekte – Energieeffizienz, Hardwareschonung und Digitale Nachhaltigkeit – einen entscheidenden Beitrag leisten.

4.5 Freie Software – Definition, Entstehung und heutige Bedeutung

Bis in die 70er Jahre des 20. Jahrhunderts wurde Software weitgehend als ein freies Gut behandelt und inklusive des offenen Quellcodes verbreitet. Computer sind zu dieser Zeit meist im universitären Umfeld zu finden. Es ist üblich, die zugehörige Software auf ihre Eigenschaften und Auswirkungen hin zu untersuchen, diese zu teilen und zu verbessern. Transparenz sowie die Wiederverwendung oder Wiederholung von Ergebnissen und Erkenntnissen bilden die Grundlagen der elektronischen Informationswissenschaft.

Mit dem Aufkommen des Heimcomputers in den 70er und 80er Jahren beginnen Hardwarehersteller jedoch damit, ihre für den Betrieb der Hardware mitgelieferte Software nur noch in verschlossenen Dateiformaten und damit ohne den zu Grunde liegenden Quellcode auszuliefern. Das Wissen um den Betrieb der Maschine soll verschlossen bleiben. 1974 wird das amerikanische C*opyright* auch auf Computerprogramme ausgeweitet. Es kommt zur Verbreitung des *Personal Computer*s der Firma *International Business Machines Corporation (IBM)* und dessen mitgeliefertem Betriebssystem *DOS* (disk operating system) der Firma *Microsoft*. Damit wird im außeruniversitären Umfeld der Grundstein zur Akzeptanz des Vertriebs *proprietärer Software* gelegt. Als proprietäre Software wird Software bezeichnet, deren Quellcode nicht ersichtlich ist und für die zudem nur eingeschränkte Nutzungsrechte gelten.

Um dieser Entwicklung gegenzusteuern, initiiert Richard Stallman – Mitarbeiter im Labor „Künstliche Intelligenz“ des Massachusetts Institute of Technology – 1983 die Entwicklung des GNU-Projektes[11]. Ziel des GNU-Projektes ist, ein komplettes Betriebssystem zu schreiben, dass vollständig aus *Freier Software* besteht. Damals bis heute wird unter Freier Software jegliche Software verstanden, die allen Nutzenden vier grundsätzliche Rechte (so genannte „Freiheiten“) einräumt[12]. Diese sind das Recht, die Software

1. zu verwenden (zu jedem Zweck, ohne Einschränkung)
2. zu verstehen (den Quellcode einzusehen, um das Programm untersuchen zu können)
3. zu verbreiten (das Programm beliebig oft zu kopieren und zu teilen)
4. zu verbessern (das Programm zu verändern und die neue Version selbst zu veröffentlichen)

Diese Nutzungsrechte werden gegeben und garantiert durch Lizenzen. Die Lizenz der Software beschreibt, was Nutzende unter welchen Umständen mit der Software machen dürfen. Im Gegensatz zu proprietärer Software räumen die Lizenzen Freier Software explizit das Recht ein, die Software zu verstehen und zu teilen. Lizenzen dieser Art gibt es viele. Die *Free Software Foundation (FSF)* führt

---

[10] Zur Erläuterung dieses Arguments siehe Abschnitt 4.9 Gemeinsame Lösungen für eine gemeinsame Ressource

[11] In Anlehnung an das damals weit verbreitete Betriebssystem *Unix* ergibt sich das rekursive Akronym *„GNU“*, das bedeutet *„Gnu's Not Unix“*.

[12] Die Anwendung und Garantie dieser vier Freiheiten für alle Nutzenden hat bedeutende Auswirkungen auf die Software selbst, deren Entstehungsprozess sowie deren Verwendung und Bedeutung. Diese Aspekte können hier allerdings nicht diskutiert werden. Für mehr Hintergrund siehe *https://fsfe.org/freesoftware*

dazu eine Liste von anerkannten Freie-Software-Lizenzen[13]. Die am häufigsten verwendeten Lizenzen[14] sind in Reihenfolge ihrer Häufigkeit *GPLv2, MIT, Apache, GPLv3* und *BSD*. Grob lassen sich diese Lizenzen in zwei „Familien" einteilen, die sich durch unterschiedliche Formen der „Weitervererbung" unterscheiden: Manche Lizenzen fordern, dass im Falle einer Veränderung oder Wiederveröffentlichung des Quellcodes der abgewandelte Code unter derselben Lizenz veröffentlicht werden muss. Damit wird erreicht, dass eine einmal als Freie Software veröffentlichte Software sowie alle Ableitungen und Weiterentwicklungen frei bleiben. Dieses weitervererbende Prinzip wird als „*Copyleft*" bezeichnet.

Andere Lizenzen hingegen fordern nicht ihre eigene Weitervererbung. Zur Abgrenzung von *Copyleft*-Lizenzen werden diese als „*permissive Lizenzen*" bezeichnet. Wer den Quellcode einer Freien Software verändert, die unter *permissiver Lizenz* veröffentlicht wurde, kann diese Abwandlung entweder unter der gleichen Lizenz veröffentlichen, unter einer *Copyleft*-Lizenz oder gar unter einer proprietären Lizenz. Beide Lizenz-Familien finden in der Praxis etwa gleich häufig Verwendung[15].

1984 kündigt Richard Stallman seinen Job, um sich in Vollzeit der Entwicklung von GNU zu widmen. 1985 gründet er die *Free Software Foundation* und entwickelt 1989 mit der Veröffentlichung der ersten *GNU General Public License (GPL)*[16] das Prinzip des *Copyleft*. Stallmans ursprüngliche Ankündigung der Entwicklung des GNU-Projektes gilt heute als Geburtsstunde Freier Software und der Freien Software Bewegung. Anfang 2000 entwickelt Lawrence Lessig aus den Prinzipien Freier Software und deren Lizenzen die *Creative Commons* Lizenzen, die eine Adoption der Ideen Freier Software auf die Verbreitung freier Wissens- und Kulturgüter darstellt. Zur selben Zeit und in den Folgejahren berufen sich weltweit immer mehr Bewegungen auf die Prinzipien Freier Software und Freien Wissens, darunter *Open Access, Open Data, Open Source, Open Knowledge* und *Open Educational Resources*.

Freie Software finden wir heutzutage in unzähligen digitalen Geräten, die uns umgeben. Darunter eingebettete Systeme (zum Beispiel Internet-Router), digitalisierte Geräte (zum Beispiel Fernseher, Kühlschrank, Mikrowelle), Taschencomputer (zum Beispiel so genannte „Smartphones"[17]), Supercomputer[18], der größte Teil der Internet-Hardware (zum Beispiel Server[19]) und mit GNU/Linux[20] auch der Laptop und Desktop. Unter Endanwendern bekannte Software ist zum Beispiel Linux, Firefox, Wikipedia, Android, Open/Libre Office, Apache, Wordpress und viele mehr.

4.6 Freie Software und ökologische Nachhaltigkeit

Die Veröffentlichung des Quellcodes, dessen Teil- und Wiederverwendbarkeit sind maßgebende Eigenschaften Freier Software. Freie Software kann durch diese Charakteristika als gemeinsame Ressource verwendet werden, wie beispielsweise das Wissen in der Bibliothek. Offenheit und Verfügbarkeit des Quellcodes bergen zudem zahlreiche positive Auswirkungen auf unsere sozialen und wirt-

---

13 Siehe *https://www.gnu.org/licenses/license-list.html*

14 Siehe Black Duck 2014

15 ebd.

16 Inzwischen gibt es eine ganze „GPL-Lizenzfamilie". Dazu gehören Versionsnachfolger (GPLv2, GPLv3) sowie Abwandlungen (LGPL, AGPL) und deren Nachfolger.

17 Das Freie Software Betriebssystem Android hat einen Marktanteil auf europäischen Mobilgeräten von 73,9%. (vgl. ZDNet 2014a)

18 97% der Top 500 schnellsten Supercomputer laufen unter Linux (vgl. ZDNet 2014b)

19 58% aller Webserver laufen mit der Freien Software Apache (vgl. W3Techs 2014)

20 Als „*GNU/Linux*" werden Betriebssysteme bezeichnet, die eine Kombination aus GNU Software und dem Linux-Kernel sind. Diese werden klassischerweise auf dem Desktop und Laptop eingesetzt.

schaftlichen Organisationsformen. Diese können hier jedoch nicht weiter ausgeführt werden; dieses Kapitel widmet sich ausschließlich den möglichen ökologischen Gewinnen Freier Software.

4.7 Modularität, Freie Software und Effizienz

Freie Software und Betriebssysteme bestehen üblicherweise aus einer Kombination einzelner Module, die mit Hilfe offener Schnittstellen miteinander kommunizieren und so zu einer Gesamtkomposition zusammengestellt werden. Diese Modularität kann dazu dienen, das verwendete Betriebssystem so schlank und effizient wie möglich den Betriebsanforderungen anzupassen. Viele Computer und Computerarbeitsplätze werden schließlich nur dazu verwendet, Texte zu schreiben und zu verarbeiten sowie diese über das Internet auszutauschen (im Folgenden „Textverarbeitungsplatz"). Wird für einen solchen Textverarbeitungsplatz ein proprietäres System „von der Stange" erworben, dann ist dieses System üblicherweise dazu ausgelegt, viele weitere Aufgaben erledigen zu können und erfordert und verwendet dazu zahlreiche Hardwarekomponenten sowie erhöhte Mindestanforderungen an die Hardware. Meist bieten die Vertreiber keine Anpassungen oder bedarfsorientierte Abstufungen ihrer Systeme an. Weil es sich um proprietäre Software handelt, haben Nutzende zudem keine Möglichkeit, das System selbst zu verschlanken oder gewisse Funktionen auszulassen. Bei Verwendung derartiger Systeme entsteht eine Herstellerabhängigkeit, in welcher der Software-Hersteller allen Nutzenden die zur Anwendung benötigte Hardware vorschreiben kann.

Bei der Verwendung eines Freien Software Betriebssystems hingegen besteht prinzipiell die Möglichkeit, jedes Modul und Programm manuell zu konfigurieren, zu entfernen, auszutauschen oder auch einzubauen. Da Freie Software geteilt werden darf, können eigens konfigurierte Systeme verbreitet werden und alle Nutzenden von dieser Konfiguration profitieren. Das führt zu Interessensgruppen – so genannte *communities* - die spezielle GNU/Linux-Software Konfigurationen – so genannte *Distributionen* – pflegen und veröffentlichen. Darunter gibt es auch Distributionen, die sich als besonders Hardware- und Ressourcenschonend hervortun[21]. Für den oben angeführten Textverarbeitungsplatz sind solche Distributionen im Sinne der Effizienz bestens geeignet. Ganz generell ermöglicht die Modularität Freier Software, ein für die jeweiligen Aufgabenbereiche zugeschnittenes System zu erstellen, ohne unnötigen Ballast. Oder, wie es Antoine de Saint-Exupéry einst ausdrückte: *"Perfektion ist nicht dann erreicht, wenn es nichts mehr hinzuzufügen gibt, sondern wenn man nichts mehr weglassen kann."*[22]

4.8 Hardware als Ressource verstehen

Die Produktion von Hardware erfordert jede Menge Ressourcen, sowohl seltene Rohstoffe als auch Energie. Jede Möglichkeit, unseren Hardwareverbrauch zu senken, kann als ein Beitrag zur ökologischen Nachhaltigkeit betrachtet werden. Darunter wäre eine bedarfsorientierte Neuanschaffung von Hardware zu verstehen sowie eine möglichst lange Verwendung alter Hardware. Wie soeben ausgeführt, bietet Freie Software, insbesondere GNU/Linux-Systeme, dafür speziell zugeschnittene Distributionen, die möglichst wenige Anforderungen an die Hardware stellen. Hinzu kommt, dass der offene Quellcode eigene Anpassungen sowie die Übertragbarkeit der Software auf verschiedene Endgeräte ermöglicht. Das heißt, für unser Beispiel des Textverarbeitungsplatzes benötige ich mit der Verwendung Freier Software nicht länger einen Laptop oder Desktop-Computer. Durch die Verwendung

---

[21] *thinkwiki.de* listet unter *„Ressourcenschonende Linux Distributionen"* (thinkwiki 2014) beispielsweise *Debian* (eine Distribution) für den Einsatz mit einer *i486*-CPU-Architektur. Das ist Hardware, die in Desktops zwischen Anfang und Mitte der 90er Jahre eingebaut wurde. In Kombination mit einer ressourcenschonenden Desktopumgebung, beispielsweise *LXDE* oder *Xfce* (siehe thinkwiki 2014), kann damit auf bereits 20 Jahre alter Hardware ein aktuelles Betriebssystem mit grafischer Oberfläche effizient betrieben werden.

[22] Saint-Exupéry 1939: S. 60

eines schlanken GNU/Linux-Systems mit grafischer Oberfläche und einer Libre-Office-Suite[23] ist ein komfortabler Textverarbeitungsplatz bereits mit viel geringerem Hardwareaufwand möglich, beispielsweise mit dem Mini-Computer *RaspberryPi*[24]. Anstatt sich also vom Hersteller diktieren zu lassen, welche Hardwareanforderungen dessen multifunktionales Betriebssystem benötigt, kann durch die Verwendung Freier Software ein möglichst schlankes System verwendet werden und in Folge dessen eine bedarfsorientierte Neuanschaffung von Hardware erfolgen.

Die Unabhängigkeit der Software und ihrer Anwendung wirkt sich zudem positiv auf den Lebenszyklus und die Langlebigkeit von Hardware aus. Mit Freier Software kann ökologisch problematischen Geschäftsmodellen entgegengewirkt werden, welche vorsehen, die Halbwertszeit von Hardware gering zu halten oder den Wert alter Hardware zu senken, indem neue Softwareentwicklungen nicht länger den Einsatz alter Hardware unterstützen. Denn selbst viele Jahre alte Rechner sind meist vollkommen ausreichend, um mit Hilfe schlanker Distributionen einen stabilen, schnellen und effizienten Textverarbeitungsplatz zu ermöglichen. Die Unabhängigkeit und Anpassungsfähigkeit Freier Software bietet dadurch die Freiheit, selbst zu entscheiden, ab wann eine Hardware zu alt für den eigenen Einsatz geworden ist.

Um den Vertrieb neuer Hardware zu fördern, wird teilweise versucht zu argumentieren, dass neue Hardware doch viel energiesparsamer sei als alte Hardware. Das ist allerdings Augenwischerei, denn die Herstellung neuer Hardware zehrt in der Energiebilanz jeden Effizienzgewinn auf. Ökologisch schneidet jede Wieder- oder Weiterverwendung von Hardware besser ab als Elektroschrott, selbst mit Recycling. Auch birgt die Verwendung alter Hardware eine soziale Komponente. Wer für den eigenen Betrieb neue Hardware benötigt, kann die ausrangierte Hardware beispielsweise noch sinnvoll als Arbeitsplatz in anderen Abteilungen, in Bildungseinrichtungen oder in der Entwicklungshilfe einsetzen.

4.9 Gemeinsame Lösungen für eine gemeinsame Ressource

Freie Software gibt jeder und jedem das Recht, die Software frei zu verwenden, zu verstehen, zu verbreiten und zu verbessern. Dadurch entsteht ein Entwicklungsmodell, welches es uns ermöglicht, Software als eine gemeinsame Ressource zu verstehen und zu verwenden. Einzelne Lösungen können so für gemeinsame Lösungen sorgen, Effizienzgewinne sich dadurch potenzieren.

Ein Beispiel: Der Stadt Berlin gelingt es, eine äußerst effiziente und maßgeschneiderte Software für ihre Verwaltung zu schreiben. Sie beschließt, diese Software per Lizenz als Freie Software zu veröffentlichen. Die Stadt Konstanz bedient sich nun dieser Freien Software und verwendet sie für ihre eigene Verwaltung. Weil die Stadt Konstanz jedoch viel kleiner ist, werden bestimmte Module nicht benötigt, zum Beispiel die Erfassung des Straßenbahnnetzes. Die entsprechenden Module werden entfernt und in Konstanz wird nun eine schlankere Version der Software zum Einsatz gebracht. Nachdem Konstanz ihre selbst angepasste Version wiederveröffentlicht, können andere Kleinstädte davon ebenso profitieren, ohne dazu erst eigene Personalressourcen einsetzen zu müssen. Schließlich fällt einem pfiffigen Mitarbeiter in Neubrandenburg beim Betrachten des Codes eine Möglichkeit auf, verschiedene Rechenprozesse zusammenzuführen und die Software damit noch effizienter zu gestalten. Nach erfolgreicher Implementierung und Wiederveröffentlichung durch die Stadt Neubrandenburg können nun auch Konstanz und Berlin die verbesserte Version des Codes in ihren eige-

---

[23] Libre Office wird hier nur als ein Beispiel für eine komplette Büroanwendungsumgebung auf Basis Freier Software verwendet. Dieselbe Aussage gilt auch für andere Büroanwendungssoftware, die als Freie Software lizenziert ist.

[24] „*RaspberryPi*“ bezeichnet einen preisgekrönten Einplatinencomputer, der besonders günstig (derzeit in etwa 40 Euro) und zudem energiesparsam ist. Seit 2012 kann dort ein komplettes Libre Office-Systen zum Laufen gebracht werden (siehe TDF 2012)

nen Anwendungsumgebungen implementieren und somit profitieren am Ende alle Anwender von der gleichen Lösung. Schließlich entsteht so eine gemeinsame Ressource und im Falle von nur einer nachhaltigen Lösung kann sich zugleich die Ökobilanz aller Anwendenden erhöhen.

Dieses Prinzip, das hier als theoretisches Beispiel angeführt wurde, finden wir in der Praxis in vielen Freie Software-Projekten[25]. Als herausragendes Beispiel sei der Linux-Kernel angeführt. Der Linux-Kernel ist eine Schnittstelle, welche die Anweisungen der Software in maschinenlesbare Anweisungen an die Hardware übersetzt. 1991 begann Linus Torvalds als einzelne Person, den Linux-Kernel zu entwickeln und als Freie Software unter der GPLv2[26]-Lizenz zu veröffentlichen. Schnell haben sich über das Internet weitere Entwickler eingefunden, um gemeinsam den Linux-Kernel zu programmieren. Heute ist der Linux-Kernel wohl der meistverwendete Kernel weltweit. Wir finden dessen Anwendung in nahezu allen Formen digitaler Geräte, vom Router zum Kühlschrank über das Smartphone und den Laptop hin zu Supercomputern. Ermöglicht wird dies durch inzwischen hunderte, tausende Programmierer weltweit, die jeder für sich immer weiter zur gemeinsamen Ressource des Linux-Kernels beitragen und diesen verbessern. Darunter sind Studierende und Freiwillige, aber vor allem auch Angestellte globaler IT-Firmen wie Intel, Red Hat, Samsung, IBM oder Google[27]. Firmen, die auf dem freien Markt in Konkurrenz zueinander stehen, erschaffen hier eine gemeinsame Ressource, die wiederum alle – inklusive der Konkurrenz – ausschöpfen können, um Kapital zu generieren. Das ist kein Widerspruch, sondern kalkuliertes Geschäft. Durch die gemeinsame Ressource vermeiden sie die Gefahr, dass Eigenentwicklungen scheitern oder obsolet werden, sie profitieren zudem von den Entwicklungen anderer und müssen nicht jeden Gedanken aufs neue erfinden oder imitieren.

Ob gewollt oder nicht, alle Beitragenden erzeugen mit der Entwicklung des Linux-Kernels eine digitale Nachhaltigkeit der gemeinsamen Ressource „Software-Kernel“. Diese zeichnet sich durch stetige Effizienzsteigerung sowie Anpassungsfähigkeit aus. Die Möglichkeit, diese Entwicklungen auf unzählige Endgeräte zu adaptieren, hat nachhaltige Auswirkungen auf die gesamte Sphäre digitaltechnischer Hardware. Positive und nachhaltige Aspekte für die Umwelt ergeben sich aus gemeinsamen Möglichkeiten der Energiesparsamkeit sowie dem Erhalt von Hardware beziehungsweise der bedarfsorientierten Neuanschaffung von Hardware.

4.10 Ausblick

Für die Beantwortung der Forschungsfrage *„Was ist Nachhaltige Software?“* beziehungsweise der Kategorisierung der Nachhaltigkeitskriterien von Software spielen Freie Software und deren Entwicklungsmodell zweifelsfrei eine wichtige Rolle. Wie gezeigt, fördert Freie Software eine digitale Nachhaltigkeit in der digitalen Gesellschaft und ist die Grundvoraussetzung dafür, Software als gemeinsame Ressource zu verstehen und zu erhalten. Ökologisch können durch Modularität und Anpassungsmöglichkeiten sowohl Energie als auch Hardware als Ressourcen geschont werden. Schließlich sorgen selbst einzelne Lösungen für gemeinsame Effizienzsteigerungen und eine gemeinsame Nachhaltigkeit. In Bezug auf die in Frage stehende Analogie zu dem „Blauen Engel“ empfehle ich deshalb eine Nachhaltigkeitskategorie „Freie Software“.

---

[25] Genau genommen finden wir das *Prinzip* überall. Allerdings sind in der Praxis manche der Projekte so klein, dass es nur eine Person ist, die tatsächlich Codezeilen beiträgt und die Software somit verbessert.

[26] Die 1991 veröffentlichte Folgeversion der ursprünglichen *Gnu General Public License*

[27] Für die Version 3.18 des Linux Kernel haben allein die oben gelisteten Firmen zusammen 29% aller Code-Änderungen beigetragen (LWN 2014)

4.11 Quellenverzeichnis

Black Duck (2014): Black Duck Knowledge Base: “Top 20 Open Source Licenses”, online: https://www.blackducksoftware.com/resources/data/top-20-open-source-licenses (abgerufen am 14.12.2014).

Brundlandt (1987): Report of the World Commission on Environment and Development (1987): “Our Common Future”. Vereinte Nationen, online: http://www.bne-portal.de/fileadmin/unesco/de/Downloads/Hintergrundmaterial_international/Brundtlandbericht.File.pdf (abgerufen am 13.12.2014).

Busch, T. (2008): Open Source und Nachhaltigkeit, in: Lutterbeck, B. / Bärwolff, M. / Gehring, R. (Hrsg.) Open Source Jahrbuch 2008, Berlin, Lehmanns Media: S. 111 – 122.

Declaration (2008): Standards and Future of the Internet. Genf, online: http://www.openforumeurope.org/library/geneva/declaration/manifesto-with-logos-final.pdf (abgerufen am 12.12.2014)

Grassmuck, V. (2004): Freie Software – Zwischen Privat- und Gemeineigentum; 2. Aufl., Bonn, Bundeszentrale für Politische Bildung.

Corbet, J. (2014): Some 3.18 development statistics; LWN.net, online: http://lwn.net/Articles/620827/ (abgerufen am 14.12.2014).

Martens, K.-U. (2013):Digitale Nachhaltigkeit, in: Kegelmann, J. / Martens, K.-U. (Hrsg.): Kommunale Nachhaltigkeit; Nomos Verlag: S. 304 – 314.

Saint-Exupéry, A. d. (1939): Terre des Hommes, III: L'Avion, S. 60, zitiert nach: https://de.wikiquote.org/wiki/Perfektion (abgerufen am 13.12.2014).

Stürmer, M. (2009): Digitale Nachhaltigkeit – ein Konzept mit Zukunft; Netzwoche #20, online: http://www.digitale-nachhaltigkeit.ch/2009/11/ein-konzept-mit-zukunft/ (abgerufen am 01.12.2014).

Thinkwiki (2014):Linux Distributionen für ältere Thinkpads; online: http://thinkwiki.de/Linux_Distributionen_f%C3%BCr_%C3%A4ltere_Thinkpads (abgerufen am 15.12.2014).

The Document Foundation, TDF (2012) LibreOffice runs on the RaspberryPi; online: http://blog.documentfoundation.org/2012/12/17/libreoffice-runs-on-the-raspberry-pi/ (abgerufen am 12.12.2014).

W3Techs (2014a): W³Techs - Web Technology Services Usage statistics and market share of Apache for websites; online: http://w3techs.com/technologies/details/ws-apache/all/all (abgerufen am 14.12.2014).

ZDnet (2014b); Beiersmann, S. (2014): iOS und Android nehmen Windows Phone in Europa erneut Marktanteile ab, ZDnet, online: http://www.zdnet.de/88209594/ios-und-android-nehmen-windows-phone-europa-erneut-marktanteile-ab/ (abgerufen am 10.12.2014).

ZDnet (2014c); Vaughan-Nichols, S. (2014): Linux dominates supercomputers as never before; ZDnet, online: http://www.zdnet.com/article/linux-dominates-supercomputers-as-never-before/ (abgerufen am 08.12.2014).

5 Nachhaltiger Betrieb von Online-Diensten

Autor: Patrik Löhr, Gründer und Geschäftsführer, Posteo e.K., Methfesselstr. 38, 10965 Berlin, presse@posteo.de

5.1 Eingangsstatement

Stehen Sicherheit und Nachhaltigkeit im Widerspruch? Nicht bei dem E-Mailanbieter Posteo.

Posteo ist ein sicherer und nachhaltiger E-Maildienst aus Berlin. Wir möchten mit Posteo einen Impuls für mehr Nachhaltigkeit und Sicherheit im Internet geben.

Die Finanzierung von Posteo erfolgt ausschließlich durch Nutzerbeiträge. Wir verwenden keine Kredite, haben keine Investoren, verdienen nichts durch Werbung, setzen nicht auf Spenden etc. Das bedeutet eine größtmögliche Unabhängigkeit.

5.2 Nachhaltigkeitskonzept

Posteo verfügt über ein umfassendes Nachhaltigkeitskonzept:

- Posteo wird zu 100% mit Ökostrom betrieben, der einen echten Beitrag zur Energiewende leistet. Das gilt nicht nur für unsere Server, sondern auch für alle Geschäftsräume.
- Unser Büro sowie unsere öffentlichen Räume, das „Posteo Lab“, sind nach Nachhaltigkeitskriterien ausgestattet (FSC-Holz aus nachhaltiger Forstwirtschaft, Recyclingpapier und -möbel)
- Unseren Mitarbeitern bieten wir einen täglichen bio-vegetarischen Mittagstisch an. Dies ist ein kostenfreies, freiwilliges Angebot für die Mitarbeiterinnen und Mitarbeiter – die dies gerne nutzen.
- Die Firmenkonten sind bei der sozial-ökologischen GLS Bank, die das Geld ausschließlich zur Finanzierung von ökologisch und sozial sinnvollen Projekten einsetzt.
- Unsere Rücklagen haben wir bei der Umweltbank angelegt. Die Bank fördert mit dem Geld Umweltprojekte
- Wir setzen im Büro und bei den Servern auf energieeffiziente und möglichst schadstoffarme Hardware, wann immer es möglich ist.
- Unsere Server-Systeme sind bedarfsgerecht dimensioniert und wachsen flexibel mit.
- Wir haben den Anspruch, stets so effizient wie möglich zu entwickeln. Durch die optimierte Gestaltung von lastrelevanten Prozessen lassen sich deutliche ressourcenschonende Effekte erzielen.
- Wir fördern Mobilität mit ÖPNV und Fahrrad, und wir geben zusätzliche Urlaubstage, wenn Mitarbeiter auf Flugreisen verzichten.

5.3 Motivation

Unsere Motivation stammt aus persönlicher Überzeugung der Gründer von Posteo, die mehrere Jahre lang ehrenamtlich im Umweltschutz engagiert waren. Wir möchten mit unserer täglichen Arbeit einen positiven Impuls für innovative, nachhaltige Konzepte im IT-Bereich geben. In der IT-Branche wird viel Greenwashing betrieben: Nachhaltige Webdienstleistungen zu erbringen, ist derzeit noch sehr schwierig. Das Internet treibt den Ressourcenverbrauch für das Wachstum aller Netzdienste und Konsumdienstleistungen an. Dramatisch wird das beim Energieverbrauch des Netzes. Gemäß einer Studie von Greenpeace International verbrauchten alleine die weltweit genutzten Cloud-Dienstleistungen im Jahr 2012 bereits mehr Energie als einzelne Industrieländer wie Deutschland oder Indien.

5.4 Nutzung von Ökostrom

Mindestens in Deutschland wird die Herkunft des Stroms, wahrscheinlich aufgrund einer gewissen Grundsensibilisierung zum Thema Klimawandel, als Problem wahrgenommen. Deshalb werben die meisten deutschen Hoster damit, ihre Dienste mit Ökostrom zu betreiben oder in irgendeiner Art „gut für die Umwelt“ zu sein. Es gibt eine Vielzahl von vorgeblich ökologischen Labeln, die das Gefühl von „Umweltfreundlichkeit“ signalisieren. Leider sind die meisten Label zwar schön, aber sie suggerieren einen ökologischen Mehrwert, den es so in der Regel nicht gibt. In Deutschland gibt es aus unserer Sicht nur vier Ökostromanbieter, die empfehlenswert sind, da nur diese Anbieter erkennbar zur Energiewende beitragen und keinen Zertifikate-Strom verwenden. Wir verwenden zum Betrieb von Posteo Strom eines Anbieters, der seinen Neukunden garantiert, nach wenigen Jahren Strom aus neu gebauten Anlagen für regenerative Energien zu liefern.

5.5 Hardware

Die Erzeugung des Stroms für die Dienste im Internet ist ein großer und wichtiger Faktor - aber nicht alles. Auch wichtig ist, wie die Hardware produziert wird, auf der die Software für die Dienstleistungen läuft. Im Bereich der Serverhardware wurden in den vergangenen Jahren von den Anbietern leider wenige Anstrengungen unternommen, auf Schadstoffe und Schwermetalle zu verzichten. Dort gab es, anders als im Bereich der „Consumer“-Elektronik, leider wenig Fortschritte. Deshalb ist es uns bei den Servern nicht immer möglich, auf nachhaltiger produzierte Komponenten zurückzugreifen - sie sind häufig nicht verfügbar. Wir bedauern das sehr und möchten uns in Zukunft verstärkt dafür einsetzen, dass sich in diesem Bereich etwas ändert. Im Büro achten wir auf die vorhandenen Öko-Label (z. B. EPEAT) bei der verwendeten Hardware - und nutzen darüber hinaus die wenigen Beispiele für faire Hardware, die es gibt (Fairphone, faire Maus). Wir versuchen auch, deren Beachtung zu fördern, indem wir sie im „Posteo Lab“ ausstellen.

5.6 Beispiele für nachhaltigere Software

Es folgen drei konkrete Beispiele aus dem Betrieb von Posteo, die die Bandbreite der Möglichkeiten in der effizienten Software-Entwicklung aufzeigen.

Posteo-Startseite

Verbesserung: Es müssen nicht immer alle Daten geladen werden!

Unsere Webseite www.posteo.de ist unser Schaufenster. Wir präsentieren Posteo und versuchen, unser Angebot Interessenten näherzubringen. Das machen wir mit viel Text - und auch mit Bildern. Posteo-Kunden, die evtl. täglich mehrfach unsere Website aufrufen, um sich einzuloggen, benötigen diese Informationen nicht immer wieder. Sie möchten sich nur einloggen - und evtl. die aktuellsten Meldungen aus unserem Blog mitbekommen.

Für diese Kunden bieten wir eine reduzierte Startseite an. Eine solche reduzierte Startseite kommt mit einem Bruchteil der Informationen der normalen Startseite aus und hat ein deutlich geringeres Ladevolumen. Es wird weniger Strom verbraucht - auf unserer Seite, und auch auf der Seite des Kunden.

Kalender

Verbesserung: Lastrelevante Prozesse durch optimierten Code reduzieren!

Vor einigen Jahren haben wir eine Kalenderfunktion eingeführt - und hierfür ein Open-Source-Produkt verwendet. In den Tests vor der Einführung war nicht aufgefallen, dass der Kalender unter

großer Last sehr ineffizient arbeiten würde. Als wir die Funktion einführten, wurden unsere Systeme jedoch umgehend einer großen Last ausgesetzt, obwohl wir damals erst wenige Tausend Kunden hatten. Und das, obwohl nur wenige Kunden überhaupt Termine eingetragen hatten. Der ursprüngliche Entwickler hatte offensichtlich nicht das Szenario eines Serverbetreibers mit tausenden Nutzern vor Augen, als er den Kalender entwickelte - sondern eher kleine Systeme, mit wenigen Nutzern. Dort funktionierte der Kalender einwandfrei. Wir mussten den Kalender also schnellstmöglich verbessern und die lastrelevanten Prozesse optimieren. Wir mussten keinerlei Funktionseinschränkungen vornehmen. Aber an sehr vielen Stellen mussten überflüssige Datenbankabfragen entfernt werden. Seit den Korrekturen läuft das System ohne nennenswerte Mehrlast mit vielen zigtausend Kunden mehr. Die Funktionalität wurde nicht verändert. Nur die Art und Weise der Programmierung wurde verbessert, um den Code sehr viel effizienter laufen zu lassen.

Beispiel Sammeldienst

Verbesserung: Nutzung geeigneter Programmiersprachen!

Ähnliches gilt für unseren E-Mail-Sammeldienst. Wir setzen dort auf eine Software, die zwar genau unseren gewünschten Zweck erfüllen konnte, nämlich E- Mails von anderen Anbietern in die Posteo-Postfächer unserer Neukunden zu transferieren. Aber auch diese Software war nicht dafür entwickelt worden, tausende Kunden gleichzeitig zu bedienen. Sie wurde mit der zunehmenden Kundenzahl immer ineffizienter, lastete immer mehr CPU-Kerne aus und holte die E-Mails in immer größer werdenden Zeitintervallen ab. Ein Zustand, den man vielleicht noch eine Weile mit immer mehr Hardware hätte kompensieren können. Oder mit besserer Entwicklung - wofür wir uns entschieden haben. In diesem Beispiel ließ sich der vorhandene Code nicht ohne großen Aufwand verbessern - deshalb haben wir uns dafür entschieden, eine komplett neue Software zu entwickeln. Diese basiert auf „Go“, einer Programmiersprache, die genau zu unserem Anliegen passt: Die neue Software kann tausende Postfächer parallel -fast in Echtzeit- auf neue E-Mails abfragen. Dabei wird nicht einmal ein CPU-Kern annährend ausgelastet - und es wird nur wenig RAM verbraucht, obwohl die meisten Verbindungen verschlüsselt sind. Für die Entwicklung effizienter Software kommt es also nicht nur darauf an, zu definieren, was eine Software leisten soll, sondern auch, unter welchen Umständen bzw. in welchem Kontext sie etwas leisten soll. Sicherheit durch Verschlüsselung wirkt sich auch nicht maßgeblich auf den Ressourcenverbrauch aus.

Noch zwei Anmerkungen am Rande: Unser Sammeldienst ist potentiell geeignet, alte Postfächer für immer bestehen zu lassen, da man sich bequem neu eingehende E-Mails von alten Postfächern „nachsenden“ lassen kann. So wäre das Postfach bei Posteo häufig nur ein weiteres Postfach eines neuen Kunden - und würde den Ressourcenverbrauch insgesamt weiter erhöhen. Deshalb schalten wir die Sammeldienste automatisiert nach drei Monaten wieder ab, um die Kunden zu einer vollständigen Migration zu Posteo zu ermutigen. Sie können den Sammeldienst zwar per Klick wieder aktivieren - wir wollen nichts vorschreiben. Aber wir wissen, dass diese Maßnahme einen Anreiz setzt, einen kompletten Wechsel zu vollziehen. Auch bleiben so keine „Prozessleichen“ bei uns im System zurück - also Sammeldienste, die auch nach Jahren noch laufen, aber schon längst keine Mails mehr transferieren, sondern nur vergessen wurden.

Und noch etwas: Bei uns können Kunden selbständig ihre Accounts kündigen, falls sie uns nicht mehr benutzen möchten - dabei werden alle Daten bei uns restlos gelöscht, um Ressourcen wieder freizugeben. Kunden haben außerdem die Möglichkeit, alle Daten selbstständig zu löschen oder zu exportieren. Da wir unabhängig und ohne Werbekunden arbeiten, müssen wir der Werbewirtschaft für ihre Werbeplätze keine „Traum-Kundenzahlen“ verkaufen und alte Accounts bestehen lassen.

Wir sind daran interessiert, möglichst viele aktive Kunden zu haben - und den Kunden, die ihr Postfach auflösen möchten, dies möglichst einfach zu ermöglichen.

5.7 Nachhaltigkeit vs. Sicherheit

Es gibt auch Bereiche, in denen sich Sicherheit und Nachhaltigkeit leider häufig noch nicht miteinander vereinbaren lassen - oder sich sogar entgegenstehen können. Vor allem deshalb, weil Hersteller es versäumen, verkaufte Geräte über längere Zeiträume hinweg weiter mit Sicherheitsupdates zu versorgen. Sicherheitsbewusste Kunden entscheiden sich spätestens dann für ein neues Gerät, wenn klar ist, dass keine Updates mehr zur Verfügung gestellt werden. Hier sollte sich dringend etwas bewegen.

Im Herbst 2014 wurde die so genannte „Poodle-Sicherheitslücke“ bekannt. Ein schon über 15 Jahre alter Sicherheitsstandard (SSLv3) galt nicht mehr als sicher und musste daraufhin abgeschaltet werden. Über SSLv3 kommunizierten u. a. einige ältere E-Mail-Programme mit unseren Servern. Alle Programme, die keine höherwertige Verschlüsselung als SSLv3 unterstützten, konnten ab diesem Zeitpunkt nicht mehr mit unseren Servern kommunizieren - also keine E-Mails mehr senden und empfangen. Das betraf nicht viele Programme, da das Nachfolgeprotokoll TLSv1 bereits seit 15 Jahren existiert. Aber: Ausgerechnet der große Hersteller Microsoft hatte mit seinen noch nicht besonders alten Windows-Phones der Versionsreihe 7.x (aus dem Jahr 2010) Probleme mit der Abschaltung. Das Telefon scheint zwar neuere Protokolle als SSLv3 zu unterstützen - aber nicht für die E-Mailprotokolle IMAP, POP3 und SMTP. Nach Bekanntwerden von „Poodle“ schalteten nicht nur wir, sondern die meisten E-Mailanbieter SSL v3 für IMAP, POP3 und SMTP für E-Mailclients ab. Daraufhin konnten zahlreiche Kunden noch vergleichsweise neuer Smartphones auf einmal ihr E-Mail-Postfach nicht mehr auf dem Handy nutzen. Die meisten Betroffenen haben uns zurückgemeldet, dass sie es gar nicht so traurig finden - und nun einen guten Grund haben, sich ein neues Smartphone zu kaufen. Schade: Wir hatten Kontakt zu Microsoft aufgenommen, leider wollten sie kein Sicherheits-Update zur Verfügung stellen. So entstehen regelmäßig eigentlich vermeidbare Berge von Elektroschrott.

5.8 Empfehlungen

- Energie, mit der die Software läuft, muss immer echter Ökostrom sein
- Hardware muss schadstoffarm und ressourceneffizient hergestellt werden
- Software im Kontext betrachten
- Software zweckgenau entwickeln
- Kompilierte Software bevorzugen
- Programmiersprachen verwenden, die in ihren Leistungsmerkmalen zur Aufgabe passen
- Aufgaben voneinander trennen, keine monolithischen Systeme
- Offene Standards nutzen, um Software austauschen zu können
- Keine Ressourcenverschwendung durch Datenleichen: Geschäftsmodell, bei dem der Nutzer das Angebot finanziert und nicht die Werbewirtschaft
- Modulare Lösungen, die Sicherheits-Updates auch noch nach Jahren zulassen

6 Diskussionsbeiträge zu Nachhaltiger Software

Für das Fachgespräch Nachhaltige Software 28.November 2014 wurden Leitfragen formuliert, die von den Teilnehmerinnen und Teilnehmern sowohl während des Fachgesprächs als auch im Nachgang beantwortet werden konnten. Hierzu wurde eine Webseite[28] erstellt, auf der die Fragen online beantwortet werden konnten. Die Antworten sollen eine Orientierung geben, welchen Geltungsbereich ein Umweltzeichen für Nachhaltige Software abdecken und welche Kriterien hierfür herangezogen werden könnten. In den folgenden Abschnitten sind die Antworten und Diskussionsbeiträge der Teilnehmerinnen und Teilnehmer zusammenfassend dokumentiert. Da es sich dabei um Antworten unterschiedlicher Personen handelt, können die Aussagen dabei widersprüchlich oder redundant sein.

Folgende Fragen wurden gestellt:

- Was ist „Nachhaltige Software“?
- Was sind mögliche Kriterien für „Nachhaltige Software“?
- Welches sind die Systemgrenzen von „Nachhaltiger Software“?
- Welche Standards und Methoden für „Nachhaltige“ bzw. „Green Software“ gibt es bereits?

6.1 Was ist „Nachhaltige Software“?

Ergänzende Fragen:

- Wie könnte eine Definition für „Nachhaltige Software“ formuliert werden?
- Welche Nachhaltigkeitsaspekte (z. B. Energie- und Ressourceneffizienz, Wirtschaftlichkeit, Sozialverträglichkeit) können dabei berücksichtigt werden?

Antworten:

- Grüne und nachhaltige Software ist Software, deren direkte und indirekte negative Auswirkungen auf Menschen, Gesellschaft und Umwelt über ihren gesamten Lebenszyklus hinweg minimal sind und die bestenfalls einen zusätzlichen positiven Beitrag zur nachhaltigen Entwicklung leistet.
- Software, die uns mit Blick auf die Ziele nachhaltiger Entwicklung voranbringt. Dies erfordert eine Definition des Nachhaltigkeits-Begriffs und der Ziele (Umwelterhalt, soziale Gerechtigkeit, Effizienz-, Suffizienz- und Konsistenz-Strategie usw.). Davon abzugrenzen wären u. a. folgende Begriffe: Langlebigkeit, System-Optimierung, digitale Nachhaltigkeit.
- Software, die Hardware-Ressourcen schont, indem ihre Erneuerung von der Hardware-Erneuerung entkoppelt ist („suffiziente Software“).
- Software, die den Nutzer in die Lage versetzt, diese im Sinne der Nachhaltigkeit zu nutzen („enabling Software“). Dies soll insbesondere durch Informationsbereitstellung an den Nutzer ermöglicht werden.
- Negativen Auswirkungen der Software sollen gegenüber dem Status Quo reduziert werden (z. B. Energieverbrauch senken) („effiziente Software“).
- Unterscheidung in:
    1. „Nachhaltige Software“ (Nachhaltigkeit der Software selbst, bezogen auf den gesamten Lebenszyklus) und
    2. „Unterstützung der Nachhaltigkeit durch Software“.
- Unterscheidung in Nachhaltigkeitseffekte 1., 2. und 3. Ordnung (direkt, indirekt, systemisch).

[28] http://oekotop100.de/software

- Unterscheidung in:
    1. Die Erstellung der Software ist nachhaltig: z. B. Module aus anderen Entwicklungen werden weiter verwendet (sofern diese für die Funktion angemessen sind). Das erstellte Softwareprodukt kann in anderen Anwendungen weiter verwendet werden. Open Source oder Open Use Software gehört dazu. Programmierer müssen fair bezahlt werden.
    2. Der Gebrauch der Software ist nachhaltig: z. B. Die Nutzung der Software darf nicht zu einer übermäßigen CPU Belastung führen.
    3. Die Software hat einen nachhaltigen Nutzen: Dies ist dann z. B. der Fall, wenn damit Umweltinanspruchnahmen (z. B. Reisen oder CO_2-Belastung) reduziert werden. Auch Software, die dazu beiträgt, dass Nutzer Erkenntnisse erlangen, die sie zu einem nachhaltigeren Verhalten veranlassen, ist nachhaltig. Letztlich ist auch jede Software nachhaltig, die direkt zum Einsparen von Umweltbelastungen beiträgt.

6.2 Kriterien für „Nachhaltige Software“?

Ergänzende Frage:

- Woran kann eine „Nachhaltige Software“ anhand eines mess- und reproduzierbaren Maßstabes erkannt werden?

Antworten:

- **Ressourcenschonung** – Entkopplung von Software-Erneuerung und dem Bedarf zur Hardware-Erneuerung.
- **Modularität** – Software sollte so aufgebaut sein, dass nur jene Module installiert oder in den Arbeitsspeicher geladen werden müssen, die für die jeweilige Anwendung tatsächlich erforderlich sind.
- **Digital Divide** – Nachhaltige Software sollte nicht der ersten Welt oder gewissen Personenkreisen vorenthalten bleiben, sondern den Zugang und die Verwendbarkeit für alle ermöglichen.
- **Software als Ressource / Wissen** – Nachhaltige Software sollte zugleich die digitale Nachhaltigkeit fördern. Dazu gehören mindestens Offene Schnittstellen, Export in offene Datenformate, Plattformunabhängigkeit und die Möglichkeit eines Systemwechsels. Proprietäre Datenformate und Vendor-lock-ins (nicht lösbare Abhängigkeiten zu einem bestimmten Anbieter) erzeugen Insellösungen anstelle einer umfassenden Vernetzung.
- **Knowledge sharing** – Frei lizensierte Software ermöglicht den Wissensaustausch über alle kulturellen und staatlichen Grenzen hinweg. “best practice” Software kann so direkt global verteilt und eingesetzt werden.
- **Schlankheit** („weniger ist mehr“)– Ähnlich der “Unix-Philosophie” die besagt
    - Schreibe Computerprogramme so, dass sie nur eine Aufgabe erledigen und diese gut machen.
    - Schreibe Programme so, dass sie zusammenarbeiten.
    - Schreibe Programme so, dass sie Textströme verarbeiten, denn das ist eine universelle Schnittstelle.
- **Plattformunabhängigkeit** – Software sollte nicht an die Verwendung bestimmter Hardware gebunden sein.
- **Offene Standards** – Die Verwendung von offenen Datenformaten sollte Bedingung für den Titel „Nachhaltige Software” sein, da alles andere Monopolstellungen fördert.
- **Dezentrale Architektur** – Einzelne Instanzen sollten dezentral laufen können, dies verringert Abhängigkeit und Monopolstellungen.

- **Öffentlicher Code** – Freie Software ermöglicht Anpassungen, Modularisierungen, vom Ersteller unabhängige Lebensdauer, Transparenz sowie Portabilität und Lokalisierungen.
- **Offline-Nutzung** – Daten sollten exportiert werden und Offline genutzt werden können. Die Software selbst sollte auch mit der lokalen Kopie der Datenbank offline nutzbar sein (vgl. z. B. OpenStreetmap und GoogleMaps – letzteres funktioniert in vielen Regionen nur online. Ersteres kann inklusive Navigationsführung komplett offline genutzt werden).
- **Unpersonalisierte Nutzung** – Es sollte möglich sein, eine Software unpersonalisiert zu benutzen (vor allem im Hinblick auf Internet of Things und Elementen wie Sprachsteuerung immer interessanter)
- **Niedriger Energiebedarf** – Da wir zurzeit keine Benchmarks haben, ist hier wichtig, dass es einen Nachweis gibt, dass der Energiebedarf der Software zumindest gemessen worden ist. Somit können später zumindest die Versionen der gleichen Software miteinander in punkto Energieeffizienz verglichen werden.
- **Geringe Belastung der Hardware** – dieselbe Problematik wie beim Energiebedarf, deshalb muss die Hardwarebelastung zumindest gemessen worden sein und die Resultate müssen leicht zugänglich sein.
- **Geringe im Netz übertragene Datenmenge** – analog Energiebedarf und Hardware-Auslastung.
- **Transparenz** (hängt mit den vorerwähnten Kriterien zusammen) – dieser Grundsatz sollte sowohl für den Code (es sollten zumindest die Schnittstellen offengelegt werden) als auch für alle Lebenszyklen der Software gelten. Z. B. Welche Ressourcen werden bei der Planung und Implementierung der Software verbraucht? Welche Ressourcen benötigt die Software, wenn sie später läuft (RAM, CPU, Netzbelastung etc.)? Welche Daten werden gespeichert und wo? Wie lange verbleiben sie da? Was genau bleibt als Rückstand bei einer Deinstallation der Software?
- **Abwärtskompatibilität** – die Kernfunktionalität einer Software ist in einem Modul verfasst, das auch auf einer älteren Hardware lauffähig ist. Ältere Hardware bedeutet hier: Solche, die vor 5 Jahren aktuell war.
- **Wissensaufbau** – die Probleme und die Lösungen, die während der Implementierung der Software aufgetaucht sind, werden festgehalten und öffentlich zugänglich gemacht. Dokumentation, Handbücher für Entwickler.
- **Möglichkeit zur Abschaltung** der nicht verwendeten Hardware.
- Unterstützung von energieeffizienten **Übertragungsprotokollen**.
- **Speichereffizienz** – nur das Notwendige speichern und automatisiert später löschen, sparsame Datenformate verwenden.
- **Verwendung offener Datenformate** – Customer-lock-ins mittels Datenformaten vermeiden.
- **Flexibilität** bezüglich Peripheriegeräten.
- **Auswahl verschiedener Energieeffizienzmodi** – Der User soll selber über Energieverbrauch vs. Performance entscheiden. Teilweise soll dies je nach Systemumgebung, in der die Software gerade läuft (PC, Tablet, LAN, WLAN…), auch automatisiert geschehen.
- **Datenschutz** – Nicht mehr erfragen als nötig, nur verschlüsselte Übertragung sensibler Daten, verschlüsselte Speicherung sensibler Daten, keine Weitergabe der Daten.
- **Minimale Datenübertragung über Mobilfunknetze** – die Software soll erkennen, wann eine Mobilfunkverbindung (LTE etc.) besteht, um dann die Datenübertragung zu drosseln oder komplett zu stoppen. Ebenso denkbar ist eine Regel, die besagt, dass Rechenoperationen, die vom Client (z. B. Tablet, Handy) angestoßen sind und im Normalfall auf dem Webserver ausgeführt werden, im Falle einer Mobilfunkverbindung auf dem Clientgerät stattfinden sollen.

- **Default-Einstellungen** (beispielsweise von Energiesparplänen) sind so gestaltet, dass nachhaltige Nutzungsmuster voreingestellt sind. Erfahrungsgemäß werden diese Default-Einstellungen nur selten geändert.
- **Monitoringtools** –Tools für den Energieverbrauch, RAM oder CPU-Beanspruchung sollen in die Benutzeroberfläche der Software intergiert werden, damit der Nutzer selbst entscheiden kann, welche Programmteile er aktiviert und wie er die Software optimal nutzen kann.
- **Unternehmenspolitik** / Corporate Social Responsibility (CSR) – Übernahme von Unternehmensverantwortung des Software-Unternehmens zur Energie- und Ressourcen-Einsparung. Beispielsweise durch die Dokumentation sinkenden Hardwareaufwands für neue Software-Releases.

6.3 Systemgrenzen von „Nachhaltiger Software“?

Ergänzende Fragen:

- Wo beginnt und wo endet der Einfluss des Softwareentwicklers / der Softwareentwicklerin? Welche Lebenszyklus-Phasen können bei der Bewertung der Nachhaltigkeit sinnvollerweise berücksichtigt werden?
- Wo sind die nicht beeinflussbaren Abhängigkeiten z. B. zu Hardware- Plattform, Entwicklungsumgebung und Nutzerverhalten?
- Welche Umweltwirkungen können einbezogen werden (z. B. auch Hardware, technischen Infrastruktur, Datenübertragung)?

Antworten:

- Zunächst sollten sämtliche Umweltwirkungen betrachtet werden, die einen Bezug zur Nachhaltigkeit haben.
- Der Softwareentwickler eines nachhaltigen Software-Produkts muss die aktuell gängige Hardware als etwas Vorhandenes annehmen. Er muss dafür sorgen, dass sein Programm mit den Ressourcen in einer vorhandenen Umgebung nicht nur zurechtkommt, sondern auch diese möglichst sparsam in Anspruch nimmt. Er übt zudem einen indirekten Einfluss auf zukünftige Versionen aus, indem er eine bestimmte Architektur wählt und den Grad der Änderungsfreundlichkeit und Anpassbarkeit festlegt.
- Die Anwendungsphase ist die entscheidende, da wir hier die Multiplikationseffekte von der Verwendung der Software haben. Umso mehr und breiter die Verwendung umso weniger fällt die Planung- und Entwicklungsphase ins Gewicht.
- Das Nutzerverhalten kann nicht beeinflusst werden. Es kann aber mit der Einhaltung von Sicherheitsstandards zumindest dafür gesorgt werden, dass die Folgen eines negativen Verhaltens seitens der Nutzer minimiert werden.
- Die Entwurfsphase ist entscheidend, da hier die Festlegung auf Software-Komponenten (Frameworks, Klassenbibliotheken, ...) erfolgt, die unnötig komplex sind und teilweise sehr hohe Hardware- und Betriebssystem-Anforderungen stellen.

6.4 Welche Standards und Methoden für „Nachhaltige“ bzw. „Green Software“ gibt es bereits?

Antworten:

Workshopserien:

- Energy Aware Software-Engineering and Development (EASED)
- Green and Sustainable Software (GREENS)

- Software Engineering Aspects of Green Computing (SEGC)

Konferenzen:

- ICT for Sustainability (ICT4S)

Richtlinien für Green Web Engineering:

- Dick, M., Naumann, S., Held, A. (2010): Green Web Engineering. A Set of Principles to Support the Development and Operation of "Green" Websites and their Utilization during a Website's Life Cycle. In: Filipe, J.,Cordeiro, J. (Hg.) (2010): WEBIST 2010 – Proceedings of the Sixth International Conference on Web Information Systems and Technologies, Volume 1, Valencia, Spain, April 07-10, 2010, 2 volumes, INSTICC Press, Setúbal, S. 48–55.
- Dick, M., Naumann, S., Kuhn, N. (2010): A Model and Selected Instances of Green and Sustainable Software. In Berleur et al. (2010) S. 248-259.

Strom-Messtool:

- Atom Leap – P. Peterson, D. Singh, W. Kaiser; Reiher, P. (2011): Investigating energy and security trade-offs in the classroom with the atom leap testbed. In 4th Workshop on Cyber Security Experimentation and Test (CSET), pages 11–11. USENIX Association, 2011.

Modelle, Prozessbeschreibungen, Indikatoren:

- Shenoy, S.S., Eeratta, R. (2011): Green software development model. An approach towards sustainable software development. In: Negi, A. (ed.): Annual IEEE India Conference (INDICON), 2011. 16 – 18 Dec. 2011, BITS Pilani, Hyderabad Campus, Hyderabad, India; proceedings, pp. 1–6. IEEE, Piscataway, NJ (2011). doi:10.1109/INDCON.2011.6139638
- Lami, G., Fabbrini, F., Fusani, M. (2012): Software Sustainability from a Process-Centric Perspective. Systems, Software and Services Process Improvement. In: Winkler, D., O'Connor, R.V., Messnarz, R. (eds.): Systems, Software and Services Process Improvement. 19th European Conference, EuroSPI 2012, Vienna, Austria, June 25-27, 2012. Proceedings, volume 301, pp. 97–108. Springer, Berlin, Heidelberg (2012)
- Albertao, F. (o.J.): Sustainable Software Engineering. http://www.scribd.com/doc/5507536/Sustainable-Software-Engineering#about
- Albertao, F., Xiao, J., Tian, C., Lu, Y., Zhang, K.Q., Liu, C. (2010): Measuring the Sustainability Performance of Software Projects. In: IEEE Computer Society (ed.): 2010 IEEE 7th International Conference on e-Business Engineering (ICEBE 2010), Shanghai, China, pp. 369–373 (2010). doi:10.1109/ICEBE.2010.26
- Mahmoud, A. (2013): A Green Model for Sustainable Software Engineering. International Journal of Software Engineering and Its Applications Vol. 7, No. 4, July, 2013
- Jiang, T., Kipp, A., Cappiello, C., Fugini, M., Gangadharan, G., Ferreira, A.M., Pernici, B., Plebani, P., Salomie, I. and Cioara, T., et al. (2014): Layered Green Performance Indicators Definition. GAMES – Green Active Management of Energy in IT Service centres, Project ICT-248514, Deliverable D2.1 WP2. http://www.green-datacenters.eu/index.php?mact=Uploads,cntnt01,default,0&cntnt01category=PublicDeliverables&cntnt01mode=single&cntnt01sortorder=name_asc&cntnt01upload_id=132&cntnt01returnid=95, 2014-10-24

7 Zusammenfassung und Ausblick

Auf dem Fachgespräch „Nachhaltige Software“ wurde anhand von Kurzvorträgen und Diskussionsbeiträgen aufgezeigt, dass Software sehr viele Nachhaltigkeitsaspekte berührt. Entlang des gesamten Lebenszyklus von Software, d. h. von der Software-Entwicklung bis zur Deinstallation, können Wirkungen auf die drei Nachhaltigkeitsdimensionen Ökologie, Wirtschaft und Gesellschaft festgestellt werden.

Bezogen auf **ökologische Aspekte** wurde dabei besonders die starke Verflechtung von Software mit Hardware-Anforderungen hervorgehoben. Die Erweiterung oder der Austausch von Computern, Servern, Speicher und Netzwerkkomponenten wird in der Regel von Software-Anforderungen getrieben. Nachhaltige Software bezeichnet in diesem Zusammenhang daher Software, die genügsam gegenüber Hardwareanforderungen ist und auch auf älteren oder schlanken Hardware-Plattformen lauffähig ist. Als weiterer wichtiger ökologischer Aspekt wurde identifiziert, dass Software dazu befähigt, Energie- und Ressourcenmanagement zu betreiben. Damit trägt eine Nachhaltige Software zur Einsparung und Optimierung bei.

Zu den **wirtschaftlichen Aspekten** von Software gehört, dass mit steigender Komplexität von Software die Abhängigkeit von wenigen spezialisierten Software-Unternehmen steigt. Die Folge sind Software-Preise, die im Oligopol festgelegt werden, Marktbeherrschung und Konzentration von vertraulichen Daten bei wenigen großen Unternehmen. Als Gegenstrategie zur Konzentration wurden im Fachgespräch *Freie* und *Open Source Software* identifiziert, die einen gleichberechtigten Marktzugang aller Softwareentwickler ermöglichen. Zusätzlich kann die Offenlegung von Software- und Hardware-Schnittstellen sowie von Datenformaten dazu beitragen, dass auch proprietäre Produkte von firmenfremden Programmierern (dezentral) weiterentwickelt und langfristig gepflegt werden.

Gesellschaftliche Nachhaltigkeit wird durch Software insbesondere mit positiven Wirkungen erreicht. Das allgemeine Bereitstellen von Wissen, Kunst und Kultur befördert die Bildung und kulturelle Teilhabe. *Freie Software* ermöglicht den Zugang aller sozialen Gruppen zu Werkzeugen und Diensten der Informationstechnik und trägt damit zur Chancengleichheit bei. Die Nachhaltigkeit einer Software kann daran gemessen werden, inwieweit durch sie solche gesellschaftlichen Aufgaben erfüllt werden.

Es wurde diskutiert, dass **Nachhaltigkeitseffekte** von Software **auf unterschiedlichen Ebenen** stattfinden können. Am leichtesten zu quantifizieren sind direkte Effekte von Software (Effekte 1. Ordnung), beispielsweise die Inanspruchnahme von CPU-Leistung, Arbeitsspeicher oder Datenübertragungs-Bandbreite. Wichtiger sind in der Regel jedoch indirekte Effekte (Effekte 2. Ordnung), indem beispielsweise Software einen Ausbau der Hardware erfordert oder dazu beiträgt, einen Produktionsprozess zu optimieren. Die stärksten Auswirkungen auf die Nachhaltigkeit werden von systemischen Effekten von Software erwartet (Effekte 3. Ordnung). Solche Systemänderungen sind beispielsweise die Änderung des Konsumverhaltens (Online-Shopping statt Einkaufsfahrten) oder Änderungen der Arbeitswelt (Home-Office statt Präsenzarbeitszeiten). Systemische Effekte können nicht einer einzelnen Software zugeordnet werden, sondern vielmehr der Digitalisierung von Lebenswelten.

Um die Nachhaltigkeitseffekte von Software zu analysieren, können grundsätzlich **bestehende Methoden** angewendet werden, wie sie bei der Analyse von physischen Produkten verwendet werden, wie die Ökobilanzierung, Impact Assessment oder Produktnachhaltigkeitsanalysen (PROSA). Allerdings gibt es dabei noch einige methodische Herausforderungen zu überwinden. Diese beginnen bei der Festlegung von Bilanzgrenzen (z. B. gehört die Hardware zur Software dazu?), gehen über die Quantifizierung von Nachhaltigkeitseffekten (z. B. wie wird kultureller Beitrag gemessen?) und reichen bis zur geeigneten Zuordnung von Energie- und Ressourcenverbräuchen zu parallel operieren-

der (Betriebs-)System-Software und der umgebenden Hardware. Bei Effekten zweiter und dritter Ordnung kommen weitere methodische Probleme der geeigneten Zuordnung von Effekten und Feststellung von Wirkungszusammenhängen hinzu.

Als **Ausblick** lässt sich deshalb festhalten, dass noch erheblicher **Forschungsbedarf** bei Nachhaltiger Software besteht. Bislang wurden Nachhaltigkeitsaspekte bei Software noch wenig erforscht, weshalb das derzeitige Fehlen an Methoden und Maßstäben an die Anfänge der Ökobilanzierung und der integrierten Produktpolitik erinnern. Die Entwicklung eines Bewertungssystems für ein Produktkennzeichen für Nachhaltige Software (z. B. ein Umweltzeichen „Blauer Engel“) könnte den Einstieg in ein besseres Verständnis der Nachhaltigkeitseffekte von Software bieten und für die Softwareentwicklung insgesamt richtungsweisend sein.

Auf der Grundlage der auf dem Fachgespräch geführten Diskussion werden folgende konkrete Forschungsinhalte zur Prüfung und Vorbereitung der Machbarkeit eines Umweltzeichens für Nachhaltige Software vorgeschlagen:

- Identifizierung der relevanten Nachhaltigkeitsaspekte, die durch Nachhaltige Software abgedeckt werden: Innerhalb aller drei Nachhaltigkeitskategorien sollten die wesentlichen Wirkungen von Software festgestellt und systematisiert werden. Die Untersuchung sollte dabei nicht auf die direkte Wirkung von Software in der Nutzungsphase beschränkt bleiben, sondern vielmehr den gesamten Produktlebenszyklus (insbesondere inklusive Softwareentwicklung) umfassen und auch indirekte sowie systemische Effekte von Software benennen.
- Entwicklung von Messmethoden und Bewertungsmaßstäben, um diese Nachhaltigkeitsaspekte zu quantifizieren: Mit Blick auf eine Optimierung der Software-Entwicklung und die Ableitung von Kriterien für ein Produktkennzeichen sollten für jeden Nachhaltigkeitsaspekt geeignete Methoden entwickelt werden, diese Aspekte zu quantifizieren. Dies können Messmethoden mit zugehörigen Prüfanordnungen sein, die reproduzierbare Messergebnisse (z. B. für den Energieverbrauch) liefern, Softwaretools zum Energie- und Ressourcen-Monitoring von Software, aber auch Selbstevaluations-Werkzeuge, wie sie beispielsweise vom Deutschen Nachhaltigkeitskodex (DNK) für die Nachhaltigkeitsberichterstattung von Unternehmen empfohlen werden.
- Analyse von bestehender Software im Hinblick auf die identifizierten Nachhaltigkeitsaspekte, Sammlung von best-practice- und worst-practice-Beispielen: Um das entwickelte Regelwerk anhand von Praxisbeispielen zu testen, sollte eine größere Anzahl an Software-Produkten analysiert werden. Best-practice-Beispiele legen dabei das Anforderungsniveau für nachhaltige Software fest, während worst-practice-Beispiele dabei helfen, mögliche negative Nachhaltigkeitseffekte zu identifizieren, die als Ausschlusskriterien festgelegt werden müssen (z. B. Programmierung in Kinderarbeit, Spionage- und Schadsoftware, Software mit Suchtpotenzial).
- Festlegung von Benchmarks und Bewertungssystemen zur Identifizierung von Nachhaltiger Software: Diese Kriterien ergeben sich aus der vorangegangenen Auswertung und können zur Entwicklung von Vergabekriterien für ein Produktkennzeichen herangezogen werden.
- Zusammenstellung von Regeln und Handlungsempfehlungen zur nachhaltigen Softwareentwicklung: Dieser Forschungsinhalt richtet sich an Softwareentwickler, die eine Nachhaltige Software entwickeln oder eine bestehende Software dahingehend optimieren möchten. Dabei sollen praxisgerechte Handlungsempfehlungen gegeben werden, die dabei unterstützen, nachhaltigere Software zu programmieren (z. B. Hinweise auf zur Verfügung stehende schlanke Bibliotheken, Optimierungswerkzeuge, Energie-, Ressourcen- und Bandbreiten-Monitoring).
- Das Ergebnis der Forschungen sollte die Konkretisierung eines Umweltzeichens für Nachhaltige Software sein und es sollte eine Handlungsempfehlung für die Politik abgeleitet werden, ob dies ein geeignetes Instrument ist, nachhaltige Produktpolitik im Bereich der Software zu betreiben.