Software Engineering und Integrative Nachhaltigkeit

Benno Schmidt, Andreas Wytzisk

Hochschule Bochum
Lennershofstr. 140
D-44801 Bochum
benno.schmidt@hs-bochum.de
andreas.wytzisk@hs-bochum.de

Abstract: Die Schaffung und Nutzung informationstechnischer Systeme geht einher mit positiven wie auch negativen Wirkungen auf Mensch, Gesellschaft und Umwelt. Ausgehend vom Konzept der Integrativen Nachhaltigkeit und einer Definition des Begriffs der Nachhaltigen Software werden die Anforderungen an nachhaltige Software-Engineering-Prozesse diskutiert. Auf Grundlage einer modellhaften Prozessbeschreibung werden mögliche Strategien identifiziert, welche eine zu den aufgestellten Zielen konforme Software-Entwicklung ermöglichen. Zuletzt werden mit Blick auf die dargestellte Problematik offene Forschungsfragen benannt.

1 Einleitung

Aspekte der Nachhaltigkeit werden heute intensiv mit Blick auf die Hardware-Komponenten informationstechnischer Systeme diskutiert (Stichwort "Green IT"). Für den Bereich der Software-Entwicklung finden in verschiedenen aktuellen Forschungsprojekten zwar verstärkt und mit Erfolg entsprechende Betrachtungen statt, jedoch ist der diesbezügliche Diskussionsprozess sicherlich noch nicht als abgeschlossen anzusehen [Ta11, Pe12, Pe13, Na13].

Zunächst lässt sich konstatieren: Die Entwicklung und Nutzung von Informations- und Kommunikationstechnik (ICT) ist grundsätzlich mit einem erheblichen Verbrauch von Ressourcen (Energie, Rohstoffe etc.) verbunden [Ga07]. Systemoptimierungen unter dem Effizienz-Aspekt sind zwar wünschenswert, lösen diese Problematik jedoch nicht abschließend: Insofern materielle Ressourcen irreversibel verloren gehen, ist ein energieeffizientes und Ressourcen-schonendes ICT-System zwar seiner ineffizienten Variante vorzuziehen, lässt sich aber sicherlich nicht allein auf Grund eines minimierten Bedarfs als nachhaltig bezeichnen. Insofern stellt sich die Frage, ob und ggf. wie sich Systeme schaffen lassen, deren umfassend betrachtete Auswirkung auf die ökologische und soziale Umwelt nicht-negativ oder sogar positiv ist. Explizit einzubeziehen in die Betrachtung sind dabei die Software-Komponenten und deren Herstellungsprozesse, die mit nennenswerten Bedarfen und Nebenwirkungen verbunden sein können.

Im Rahmen des im Aufbau befindlichen Forschungsschwerpunktes "Nachhaltige Entwicklung" an der Hochschule Bochum wird mit dem Nachhaltigkeitsziel auch eine gerechte Verteilung der Ressourcen verbunden. Eine zentrale Herausforderung stellt hierbei die globale Verbesserung der Lebensbedingungen für alle Menschen dar. Dies führt zu einer im Umfeld von Software-Entwicklungen zusätzlich zu beachtenden Anforderung.

Ausgehend vom Konzept der Integrativen Nachhaltigkeit (Kapitel 2) wird im Weiteren zunächst diskutiert, was sich unter nachhaltiger (im Gegensatz zu langlebiger oder effizienter) Software und unter nachhaltigen Software-Engineering-Prozessen verstehen lässt (Kapitel 3). Auf Grundlage einer modellhaften Prozessbeschreibung werden daraufhin Strategien abgeleitet, welche eine mit den Zielen vereinbare Software-Entwicklung erlauben, und praktische Beispiele genannt (Kapitel 4).

2 Nachhaltigkeitsbegriff

2.1 Integrative Nachhaltigkeit

Nachhaltigkeit lässt sich auf verschiedene Art und Weise definieren [Do14]. Häufig wird aus dem Leitbild der "Brundtland-Kommission" für Umwelt und Entwicklung der Vereinten Nationen zitiert [Wo87]: *"[Sustainable development] meets the needs of the present without compromising the ability of future generations to meet their own needs."* Im Rahmen der an der Hochschule Bochum durchgeführten Aktivitäten wird auf eine Definition Bezug genommen, die hierüber und auch über das "Drei-Säulen-Modell" mit seiner ökologischen, ökonomischen und sozialen Dimension hinaus geht. Es wird dabei an die vier Qualitäten der Pseudo-, schwachen, starken und Ultra-Nachhaltigkeit, die Martens und Schilder [MS12] aus ihrer politikwissenschaftlichen Perspektive darstellen, angeknüpft. Der dortige Begriff der Ultra-Nachhaltigkeit wurde hier leicht modifiziert und um psychologische und umweltwissenschaftliche Aspekte ergänzt, um schließlich zur Integrativen Nachhaltigkeit zu gelangen, welche sich durch folgende Merkmale auszeichnet [Sc13]:

- Verfolgt werden die Effizienz-, Konsistenz- und Suffizienz-Strategie (wie bei der starken Nachhaltigkeit). Die *Effizienz-Strategie* fokussiert auf die Minimierung des für eine Leistung zu erbringenden Aufwands, die *Konsistenz-Strategie* fordert eine Kreislaufführung der eingesetzten Stoffe und Energie. Die *Suffizienz-Strategie* zielt auf eine Änderung des Nutzungsverhaltens und der Nutzungsbedürfnisse ("nur das benutzen, was wir wirklich brauchen").
- Darüber hinaus werden eine Einheit von Mensch, Gesellschaft und Natur sowie der Gedanke der "fully-functioning society" zu Grunde gelegt. Somit wird davon ausgegangen, dass Menschen in der Lage sind, sich als Individuum zugehörig zu einer größeren Weltgemeinschaft zu empfinden.
- Naturerhalt und soziale Gerechtigkeit werden als die beiden wesentlichen Aspekte nachhaltiger Entwicklung angesehen.

2.2 Digitale Nachhaltigkeit

Im Umfeld der ICT häufig verwendet wird daneben der Begriff der Digitalen (oder Informationellen) Nachhaltigkeit, für den mehrere verschiedene Definitionen (bzw. verschiedene mit dem gleichen Begriff belegte Konzepte) existieren. Zumeist wird Wissen hier als immaterielle Ressource aufgefasst und der Nutzungsaspekt von Information inklusive offener Zugangsmöglichkeiten in den Vordergrund gestellt [Wi14, Bu08]. Insofern vernachlässigt diese Sicht (die stark die Open-Source-Philosophie stützt) mit Blick auf unsere Nachhaltigkeitsdefinition zunächst die ökologische Dimension, indem nicht explizit auf die Schonung natürlicher Ressourcen und nicht auf die oben geforderte Effizienz und Konsistenz abgezielt wird [Ma13]. Die weiteren Ausführungen werden zeigen, dass ungeachtet dessen das Software-gebundene Wissen (aus einer semiotischen Sichtweise heraus) positiv bilanzierbar sein kann. Weiterhin sind der offene Zugang zu Software-Artefakten und Wissen sowie die Möglichkeit der Partizipation an den Entwicklungsprozessen positiv hervorzuheben. Insofern vermag das Konzept der Digitalen Nachhaltigkeit verschiedene Teilaspekte Integrativ-nachhaltiger Software-Entwicklung anzusprechen.

3 Begriff der Nachhaltigen Software

In Anlehnung an den in der Literatur recht breit akzeptierten Vorschlag von [Di10] soll nachstehende Definition als erster Ausgangspunkt für die nachfolgenden Überlegungen dienen[1]: *(Integrativ-) Nachhaltige Software zeichnet sich dadurch aus, dass die direkten und indirekten negativen Auswirkungen auf Mensch, Gesellschaft und Umwelt, die sich aus der Entwicklung, dem Betrieb und der Verwendung der Software ergeben, minimal sind. Zudem sollen sich mit Blick auf eine (integrativ-) nachhaltige Entwicklung durch die Software langfristig positive Auswirkungen ergeben.*

Unter "langfristig positiven Wirkungen" lässt sich mit der Zielsetzung einer Integrativen Nachhaltigkeit eine Entwicklung verstehen, die den Einklang von Mensch, Gesellschaft und Natur im Sinne eines "bien vivir" unter Einhaltung natürlicher Stoffkreisläufe und frei von irreversiblen Änderungen anstrebt; siehe auch [Sc13]. Technologie im Allgemeinen und insbesondere auch ICT-Systeme werden dabei im Rahmen des genannten Forschungsschwerpunktes als treibender Faktor zur Entwicklung von Hilfsmitteln und Vorgehensmodellen zur Unterstützung und Förderung nachhaltiger Lebensweisen angesehen.[2]

4 Nachhaltige Software-Entwicklung

Oben stehende Definition für Nachhaltige Software beinhaltet bereits den Entwicklungsprozess. Trotzdem sei an dieser Stelle eine abstrakte Sicht auf unser Tun als

[1] Die gegenüber [Di10] vorgenommenen Änderungen sind in [Sc14] erläutert.
[2] An dieser Stelle wird marginal von der Ultra-Nachhaltigkeitssicht abgewichen; [MS12] spricht eine "radical critique of Western culture, including its anthropocentrism, belief in science and technology […]" an.

Software-Entwickler gewagt, welche als Grundlage für die Ableitung eines einfachen Prozessmodells dienen soll.

4.1 Software Engineering als Transformationsmodell

In einer naiven Anschauung lässt sich die Entwicklung und die sich anschließende Nutzung von Software prinzipiell als eine Transformation von (individuellem) menschlichem Wissen w und Daten d (worunter gemäß dem Von-Neumannschen Konzept auch Software-Artefakte, insbesondere Computer-Programme fallen) in neues Wissen w_t (u. a. Handlungswissen) und weitere Daten d_t, u. a. Software, betrachten. Stützen lässt sich diese Sichtweise z. B. durch die im Umfeld der semiotischen Forschung aufgebauten Modelle, die Software-Entwicklung als einen auf semiotischen Entitäten stattfindenden Prozess von Zeichen-Transformationen (d_t) sowie Bedeutungs- und Wissenszuordnungen verstehen [Na07].

Dieser sozio-technische Transformationsprozess, der mit einem Austausch von Mensch, Gesellschaft und natürlicher Umwelt einher geht, bedarf der Zufuhr geeigneter materieller (inkl. energetischer) Ressourcen *r,* die im Rahmen der Entwicklung und des Betriebs der Software ebenfalls transformiert werden [RB11, Na07, Ta11], z. B. für die Hardware benötigte Rohstoffe und elektrische Energie in CO_2, Abwärme, Computer-Müll usw. Teilweise werden diese materiellen Ressourcen irreversibel umgesetzt, d. h. es ergibt sich ein Verlustanteil ε.

Nun vermag Software innerhalb ihres Lebenszyklus als Mittel menschlichen Handelns zu fungieren [WF89] und bei den Nutzern Änderungen im Nachhaltigkeitsbewusstsein bewirken, was nachfolgend symbolisch als ψ-Komponente berücksichtigt ist:

$$w\,,\, d\,,\, \psi \;\rightarrow\; w_t\,,\, d_t\,,\, \psi_t$$

$$r \;\rightarrow\; r_t + \varepsilon$$

Die erste Transition [Sc14] erfasst immaterielle, prinzipiell erneuerbare "Wissensgüter". Die zweite Gleichung, in dessen Rahmen die Gegebenheiten der Erhaltungssätze aus der Physik zu beachten sind, bezieht sich auf die materiellen Ressourcen, für die seitens der Integrativen Nachhaltigkeit eine Einhaltung natürlicher Kreislauf-Eigenschaften gefordert wird (Konsistenz-Aspekt). Nun lässt sich formulieren: Der Wissens- und Bewusstseinsgewinn $\Delta w = w_t - w$, $\Delta\psi = \psi_t - \psi$ sollte mit Blick auf die Nachhaltigkeitsziele mindestens so groß sein wie der irreversibel transformierte Anteil der eingesetzten materiellen Ressourcen, also

$$\beta(\Delta w) + \beta(\Delta\psi) - \beta(\varepsilon) \;\geq\; 0,$$

wobei β eine geeignete Bewertungsfunktion bezeichnet. Das Potenzial einer pragmatischen Nutzung des "Wissensgutes Software", welches die Anwendungskompetenz des Nutzers voraussetzt, ist hierbei der Komponente w zugeschlagen (Daten und Software

sind ohne entsprechendes Wissen nicht nutzbar!).[3] Es lässt sich argumentieren, dass den generierten Software-Artefakten neu gewonnenes Wissen inhärent ist ("computation as semiosis" bei [Na07]), wobei zu gewährleisten ist, dass dieses positive Δw durch den Menschen erschließbar bleibt. Der Aspekt der Regenerativität des immateriellen Gutes "Software" wird insbesondere im Umfeld der bereits angesprochenen Digitalen Nachhaltigkeit hervorgehoben. Angesichts der Diskussion der gegenwärtigen Umweltprobleme sei an dieser Stelle allerdings nochmals die Notwendigkeit einer Dematerialisierung und Dekarbonisierung der Software-Entwicklungsprozesse konstatiert.

Der durch eine Software-Anwendung erreichbare Wissens- und Bewusstseinsgewinn, welcher der Erreichung der Nachhaltigkeitsziele dienlich sein soll, wird nur selten so offensichtlich sein, als dass kein Gegeneinanderabwägen der Gleichungsgrößen erforderlich wäre. (Die quantitative Erfassung der Größen mittels β ist dabei schwierig. Zudem sind letztlich "Äpfel und Birnen" abzuwägen.)

Ungeachtet der Validität des aufgebauten Modells eröffnet das Vorhandensein einer formalisierten Modell-Beschreibung die Möglichkeit einer sozialen Aneignung der dahinter liegenden Prozesse [Hü85] sowie die Ableitung grundsätzlicher Strategien zur nachhaltigen Software-Entwicklung, u. a.:

- *S1* Das gewonnene Wissen wird mit Blick auf die Ziele nachhaltiger Entwicklung maximiert (fokussiert das Δw).
- *S2* Die (irreversibel) eingesetzten Energiemengen und Ressourcen werden minimiert (minimales ε; Effizienzaspekt).
- *S2′* Es werden regenerative Energiequellen für Software-Entwicklung und -Betrieb genutzt (Energie-Anteil in ε soll gegen 0 streben).
- *S3* Es werden langlebige Software-Bausteine geschaffen (Reduzierung von ε durch längere Produkt-Lebenszyklen).
- *S4* Es wird ein breiter Zugang zu den geschaffenen Software-Artefakten und dem generierten Wissen gewährleistet (Vergrößern des Δw durch Multiplikatoren; Effizienz und soziale Gerechtigkeit).
- *S5* Es werden neue Entfaltungspotenziale des Integrativen Nachhaltigkeitswissens und von Nachhaltigkeitsbewusstsein ($\Delta\psi$) geschaffen.

Nicht explizit modelliert wurde hier der Aspekt der sozialen Gerechtigkeit. Diese *wesentliche Randbedingung* ist bei der Optimierung der oben angegebenen Zielgröße zu beachten, wobei "Gerechtigkeit" ein komplexer Begriff ist.[4]

[3] Hier wäre eine differenziertere Sicht auf Wissen möglich, z. B. durch eine Unterscheidung von explizitem und implizitem Wissen ("tacit knowledge").
[4] Wir tendieren zu dem Kriterium einer bedarfsbezogenen Verteilung.

4.2 Praktische Beispiele

4.2.1 "Software Engineering for the Planet"

Eine Software kann nachhaltig sein, insofern sie eine mit Blick auf die Nachhaltigkeitsziele sinnvolle Funktion hat, z. B. indem sie die praktische Durchführung einer Tätigkeit ermöglicht, welche einen direkten positiven Effekt auf die Umwelt hat (*"Software Engineering for the Planet"* [Ta11], Strategie *S1*). Genannt seien nur einige wenige Beispiele (die Liste ließe sich lang fortsetzen):

- eine E-Government-Anwendung (digitale Formulare per Internet statt Papier und Fahrt zum Amt),
- GIS-Werkzeuge zur Bearbeitung nachhaltigkeitsrelevanter georäumlicher Fragestellungen (z. B. die Optimierung des Standorts einer Windkraftanlage; für weitere Beispiele siehe [Sc13]),
- Umgebungen, welche die Partizipation an hinsichtlich der Nachhaltigkeitsziele relevanten gesellschaftlichen und politischen Prozessen oder an Forschungsaktivitäten im Sinne einer "Citizen Science" ermöglichen [JM13, Wy13].

Software vermag allerdings auch indirekt positiv zu wirken, beispielsweise indem sie dem Menschen die Handhabung komplexer Theoriegebilde ermöglicht oder der Entwicklung von Software-Komponenten dient, welche zukünftig direkte positive Effekte auf die Umwelt haben.

4.2.2 Organisatorische Aspekte

Gestaltungsspielraum besteht bei der Organisation der Arbeitsprozesse und des operationellen Betriebs. Nachstehende Beispiele tendieren in Richtung unseres Kriteriums, wobei die Strategie *S2* (bzw. *S2'*) verfolgt wird:

- In der Anwendung ist konzeptionell ein Mechanismus verankert, der zu einer *Kompensation negativer Umweltauswirkungen* führt. Beispiel: Durch die Nutzung der Anwendung werden automatisch Geldbeträge an nachhaltige Projekte gespendet.
- Unnötige Mobilität der Projektbeteiligten wird vermieden (Energieeffizienz); Telekommunikations- und Internet-Technologien werden weitestmöglich zur "Raumüberwindung" genutzt, insofern eine Face-to-face-Kommunikation verzichtbar ist.
- Im Rahmen der Software-Entwicklung benötigte Hardware-Ressourcen (z. B. Drucker, Server, Cloud-Computing-Systeme etc.) werden von mehreren Nutzern geteilt.
- Für Entwicklung und Betrieb der Anwendung ist ausschließlich eine *Nutzung regenerativer Energie* vorgesehen (unter Einhaltung geschlossener materieller Stoffkreisläufe). Beispiel: Nutzung erneuerbarer Energien im Softwarehaus und für Elektroautos der Mitarbeiter für die Fahrten zur Arbeitsstätte; intensive Nutzung von Green IT.

4.2.3 Unterstützung in den verschiedenen Entwicklungsphasen

Wesentlich schwieriger zu beantworten ist die Frage, wie sich die praktische Software-Entwicklung während der verschiedenen, ohnehin stark von Wirtschaftlichkeits- und Effizienz-Gedanken getriebenen Entwicklungsphasen (Anforderungsanalyse, Entwurf, Implementierung etc.), nachhaltig gestalten lässt. An dieser Stelle seien nur wenige mögliche Maßnahmen, deren Sinnhaftigkeit im Einzelfall durch die Software-Entwickler zu prüfen ist, beispielhaft aufgeführt. Die meisten (in Softwarehäusern häufig genannten) Beispiele der Auflistung tragen primär zu einer erhöhten *Effizienz* (Strategie *S2*) und größtenteils auch der *Langlebigkeit* (*S3*) von Software-Lösungen bei, scheinen also eher mit Blick auf eine schwache Nachhaltigkeit (Stichwort "Effizienz-Revolution" bei [MS12]) positiv bewertbar.

- Während der Anforderungsanalyse gibt es weitreichende Partizipationsmöglichkeiten für alle Interessensbeteiligten (Stakeholder) sowie für weitere potenzielle Akteure; siehe dazu auch [Mh13].
- Einzelne Systemteile (z. B. in Dienstketten einbindbare, generisch verwendbare Web-Dienste oder breit nutzbare Datenbank-Inhalte) lassen sich auch in Fremdsystemen oder späteren Neuentwicklungen direkt verwenden.
- Die wesentlichen konzeptuellen Elemente des Software-Entwurfs und der Implementierung werden kommuniziert (Wissenstransfer unter Software-Entwicklern statt "tacit developer knowledge"; erleichterte Erweiterbarkeit von Anwendungen um neue Funktionalität; Unterstützung weiterer kreativer Wertschöpfung etc.).
- Ein System bietet langlebige (offene) Interfaces an, wodurch nach Änderung der Implementierung (z. B. System-Update) die Applikationen, die diese APIs nutzen, weiterhin lauffähig sind.
- Web-Dienst-Schnittstellen werden nicht plötzlich (proprietär) geändert, so dass darauf aufbauende Anwendungen lauffähig bleiben.
- Eine von zahlreichen Anwendungen genutzte Software-Schnittstelle ist nicht unnötig komplex, so dass eine Inwertsetzung ohne zu großen Aufwand möglich ist (vgl. z. B. [Wy13] oder INSPIRE-Beispiel bei [JM13]).
- Eine millionenfach aufgerufene Funktion (z. B. innerhalb einer Massenanwendung mit sehr großer Nutzerzahl) arbeitet algorithmisch effizient, so dass CPU-Last und Energieverbrauch reduziert werden.

4.2.4 Software-Zugang

Strategie *S4* verfolgt die Gewährleistung eines möglichst breiten Zugangs zur Software und weitergehend zu den geschaffenen Software-technischen Artefakten und dem zugehörigen Entwickler- und Nutzungswissen. Beispiele:

- Der Benutzerschnittstellen-Entwurf berücksichtigt eine breite Software-Nutzung, u. a. durch technikferne Zielgruppen, für ältere Menschen, barrierefrei, in anderen Kulturkreisen etc.

- Die zu entwickelnde Anwendung stellt keine unangemessen hohe Hardware-Voraussetzungen und zwingt den (u. U. finanziell schlecht gestellten) Anwender nicht zum Kauf eines neuen Computers, obwohl der "alte" ansonsten noch gut laufen könnte.

Unter dem Aspekt der sozialen Gerechtigkeit ist u. a. die Gewährleistung gleichberechtigter Zugangsmöglichkeiten zu Informationsressourcen und Software von Allgemeininteresse positiv bewertbar. Hierunter fällt insbesondere folgende Maßnahme:

- Es wird ein freier Zugang zu den geschaffenen Software-Artefakten wie Quellcode ("Open Source"), Schnittstellen-Beschreibungen etc. gewährt.

Angemerkt sei an dieser Stelle, dass Open-Source-Software positive Beiträge zur Erreichung des Nachhaltigkeitsziels leisten kann, es aber nicht per se tut [Ma13]. Hingegen können auch kostenpflichtige Angebote oder proprietäre "Closed-Source"-Software (die häufig als Ideengeber und treibende Kraft für innovative Software-Anwendungen fungiert) durchaus sehr nachhaltig sein.

Kontrovers diskutiert wird die Rolle mobiler Web-Anwendungen, welche möglicherweise als wesentliches unterstützendes Element für den Übergang in nachhaltigere Lebensweisen fungieren können (siehe z. B. Diskussion zur Dekarbonisierung und Dematerialisierung durch Vernetzung auch in strukturschwachen Regionen der Welt, oder Beispiele aus dem ICT4D-Umfeld). Auf Grund des hohen infrastrukturellen Wertes mobiler Web-Technologie und der dadurch geschaffenen breiten Zugangsmöglichkeiten (und mit Blick auf die heutige Lebenswirklichkeit) sei die Auflistung daher um folgenden Punkt ergänzt:

- Die Anwendung ist lauffähig auf mobilen Endgeräten, wobei auch geringe Bandbreiten und/oder Offline-Betrieb unterstützt werden sollten.

5 Fragestellungen aus Sicht der Angewandten Forschung

Im Software-Engineering-Kontext bleiben zahlreiche Fragestellungen, die es im Rahmen einer (eher trans- als interdisziplinär ausrichtbaren) angewandten Forschung unter der Zielsetzung der (Integrativen) Nachhaltigkeit zu untersuchen gilt; vgl. auch [Pe12, JM13, Ta11, HL11, Na13].

- Wie groß ist der energetische und materielle Bedarf zur Durchführung von Software-Projekten inklusive sämtlicher "Nebenkosten" (Mobilität, Bürobedarf, sich anschließender produktiver Betrieb etc.) und wie lässt sich dieser Bedarf sinnvoll dem erzielbaren Software-Nutzen gegenüberstellen? Wie lässt sich der Grad der Nachhaltigkeit von Software und Software-Entwicklungsprozessen auf praktikable Art und Weise umfassend bewerten?
- Es fehlen praktisch nutzbare Entscheidungshilfen für die Auswahl alternativ einsetzbarer Software-Anwendungen und -Komponenten, Architekturen, Entwurfsmuster und Algorithmen mit Blick auf die Nachhaltigkeitsziele.

- Welche konkreten Inhalte könnte eine Zusammenstellung nachhaltiger Verhaltensweisen im Sinne selbstverpflichtender Gesten ("mit Δψ-Effekt") für Software-Entwickler haben?
- Wie ist ein Vorgehensmodell auszugestalten, das die projektbegleitende Bewertung der Nachhaltigkeit des Entwicklungsprozesses erlaubt und im Projektverlauf entscheidungsunterstützend wirken kann?
- In der Software-Entwicklung sind sehr häufig *Rebound-Effekte* beobachtbar (Zugewinne an Performanz, Speicher etc. werden sogleich ausgenutzt)! Welche Software-Entwurfsprinzipien lassen sich aus einem tieferen Verständnis dieses Zusammenhangs heraus gewinnen?
- Wie lassen sich Software-technische Artefakte durch Nicht-IT-Fachleute (in der Westlichen Kultur ebenso wie in Entwicklungsländern) einfacher nutzbringend verwenden?

Unabhängig vom Herstellprozess ergeben sich weitere Fragen, u. a.: Auf welche Weise können mobile Web-Anwendungen als wesentliche unterstützende Elemente für den Übergang in nachhaltigere Lebensweisen dienen (in allen Regionen dieser Welt)? Welche konkreten Anforderungen ergeben sich an nachhaltige Software mit Blick auf die Themen Privatsphäre, Datenschutz, Barrierefreiheit und soziale Gerechtigkeit? Welche Anforderungen ergeben sich (über den Software-Aspekt hinaus) für eine nachhaltige Fachdatenerfassung, -haltung und -bereitstellung?

Darüber hinaus führt der vorliegende Beitrag auch zu Fragestellungen, die primär aus sozialwissenschaftlicher Sicht zu untersuchen sind, z. B.: Wie lässt sich Nachhaltigkeitsbewusstein (ψ-Komponente) auf praktikable Art und Weise aussagekräftig messen?[5] Welche Abhängigkeiten bestehen zwischen Software-Herstellung, Software-Nutzung und global gerechter Ressourcen-Verteilung? Wie lässt sich die Kluft zwischen Software- und Wissensproduktion und gesellschaftlichem Handeln in Richtung nachhaltiger Lebensweisen überwinden?[6]

6 Fazit

Mit Blick auf das Ziel einer Integrativen Nachhaltigkeit ist der Begriff der Nachhaltigen Software nicht gleichbedeutend mit dem der effizienten, langlebigen oder frei zugänglichen Software. Vielmehr werden u. a. auch Aspekte der Konsistenz sowie die sich ergebenden sozialen Kontexte umfassend zu beachten sein.

Auf Grundlage eines vereinfachten, interdisziplinären Transformationsmodells, das Betrachtungselemente aus den Bereichen Semiotik, Informatik, Physik, Psychologie und dem sozialwissenschaftlichen Umfeld beinhaltet, wurden mögliche Strategien benannt, welche eine Software-Entwicklung konform zu den aufgestellten Nachhaltigkeitszielen

[5] Ein Instrument zur Durchführung derartiger Messungen wird derzeit im Labor für Nachhaltige Entwicklung (LaNE) der Hochschule Bochum entwickelt.
[6] Hier geht es um "action research" im Sinne von [SP12].

ermöglichen. Der vorgestellte Modellansatz kann hierbei lediglich als ergänzende Sichtweise auf die dargestellte vielschichtige Problematik dienen.

Bewertungen der Nachhaltigkeit einer Software oder eines Entwicklungsprozesses werden sich nicht auf einzelne Maßnahmen oder Software-Merkmale beziehen. Vielmehr muss die Gesamtbilanz positiv im Sinne der angestrebten nachhaltigen Entwicklung sein. In diesem Zusammenhang werden Methoden zur Bewertung der Nachhaltigkeit von Software und Software-Engineering-Prozessen ebenso wie praktisch nutzbare Entscheidungshilfen für den Software-Entwickler benötigt (mögliche Anhaltspunkte geben [In11, Ro12, EC13]). Zu berücksichtigen ist hierbei auch die zeitliche Komponente: So könnte beispielsweise eine heute als nachhaltig eingestufte Maßnahme fortschrittsbedingt morgen nicht mehr als nachhaltig gelten. Und auch Software-Artefakte unterliegen durchaus einem zeitlichen Werteverfall.

Eine effektive Verwendbarkeit der geschaffenen Software-Artefakte ist weiterhin anzustreben. Ziel weiterer Aktivitäten im Umfeld nachhaltiger Software-Entwicklung sollte u. a. eine Sensibilisierung der verschiedenen Akteure (Entwickler, Projektleiter, Kunden, Anwender, Entscheidungsträger etc.) für dieses global wichtige Thema sein.

Danksagung

Der vorliegende Text ist entstanden im Umfeld des Projekts "Erlebnisraum Nachhaltige Entwicklung (ENE)", das vom Ministerium für Innovation, Wissenschaft und Forschung (MIWF) des Landes Nordrhein-Westfalen im Rahmen des Förderprogramms "FH Struktur" gefördert wird. Projektziel ist u. a. die Entwicklung einer Software mit positivem $\Delta\psi$-Beitrag. Der Dank der Autoren gebührt Petra Schweizer-Ries, Oliver Stengel und vielen Nichtgenannten für die zahlreichen Anregungen und Anmerkungen.

Literaturverzeichnis

[Bu08] Busch, T.: Open Source und Nachhaltigkeit. Open Source Jahrbuch 2008, TU Berlin.

[Di10] Dick, M.; Naumann, S.; Kuhn, N.: A Model and Selected Instances of Green and Sustainable Software. Proceedings of the 9th IFIP TC 9 and 1st IFIP TC 11 International Conference, Brisbane, Australia, Sept. 2010; pp. 248-259.

[Do14] Dosch, K.; Sachsen, K. et al.: Lexikon der Nachhaltigkeit. Aachen: Aachener Stiftung Kathy Beys; http://www.nachhaltigkeit.info (Aufruf April 2014).

[EC13] European Commission (DG Comm. Networks, Content & Technology), eds.: ICT footprint : Pilot testing on methodologies for energy consumption and carbon footprint of the ICT-sector, final report, 2013; http://www.ecofys.com/en/publication/ict-footprint

[Ga07] Gartner, Inc.: Gartner Estimates ICT Industry Accounts for 2 Percent of Global CO2 Emissions. Press Release, Stamford, CT, April 26, 2007; http://www.gartner.com/it/page.jsp?id=503867 (Aufruf April 2014).

[HL11] Hilty, L. M.; Lohmann, W.: The Five Most Neglected Issues in "Green IT". CEPIS Upgrade, XII(4), Oct. 2011; pp. 11–15.

[Hü85] Hülsmann, H.: Die technologische Formation - oder lasset uns Menschen machen. Berlin: Verlag Europäische Perspektiven, 1985.

[In11] International Organization for Standardization (ISO): ISO 14000 family (environmental management), 2004-11; http://www.iso.org/iso/iso14000

[JM13] Johann, T.; Maalej, W.: Position Paper: The Social Dimension of Sustainability in Requirements Engineering. Proceedings of the 2nd International Workshop on Requirements Engineering for Sustainable Systems, Rio, Brasil, July 15, 2013.

[Ma13] Martens, K.-U.: Digitale Nachhaltigkeit. In (Kegelmann, J.; Martens, K.-U., Hrsg.): Kommunale Nachhaltigkeit, Nomos-Verlag, 2013; S. 421-428.

[Mh13] Mahaux, M.: Could Participation Support Sustainability in Requirements Engineering? Proceedings of the 2nd International Workshop on Requirements Engineering for Sustainable Systems, Rio, Brasil, July 2013.

[MS12] Martens, J.; Schilder, K.: Sustainable Development. In (Krieger, J., ed.): The Oxford Companion to Comparative Politics, 2nd ed., Oxford, 2012; pp. 813-815.

[Na07] Nadin, M.: Semiotic Machine. Public Journal of Semiotics, I(1), Jan. 2007; pp. 57-75.

[Na13] Naumann, S.; Kern, E.; Dick, M.: Classifying Green Software Engineering - The GREENSOFT Model. 2. Workshop "Energy Aware Software-Engineering and Development" (EASED@BUIS), Oldenburg, Germany, April 2013.

[Pe12] Penzenstadler, B.; Bauer, V.; Calero, C.; Franch, X.: Sustainability in Software Engineering: A Systematic Literature Review. Evaluation & Assessment in Software Engineering (EASE 2012), Proceedings, Ciudad Real, Spain, May 2012; pp. 32-41.

[Pe13] Penzenstadler, B.: What does Sustainability mean in and for Software Engineering? Proceedings of the 1st International Conference on ICT for Sustainability (ICT4S), 2013.

[RB11] Radermacher, F. J.; Beyers, B.: Welt mit Zukunft : Die ökosoziale Perspektive. 7. Aufl., Hamburg: Murmann, 2011.

[Ro12] Rodriguez, A.: An Assessment Technique for Sustainability: Applying the IMAGINE Approach to Software Systems. Technical Report, Institut für Informatik der TU München. http://mediatum.ub.tum.de/doc/1126058/1126058.pdf

[Sc13] Schweizer-Ries, P.: Sustainability Science and its Contribution to IAPS: Seeking for Integrated Sustainability. IAPS Bulletin, 40, Autumn 2013; pp. 9-12.

[Sc13] Schmidt, B.: Geoinformatik-Ausbildung im Studiengang "Nachhaltige Entwicklung" der Hochschule Bochum. Hochschule Bochum, Geoinformatik-Bericht Nr. 14-01, 2014. Abrufbar unter http://www.geoinformatik-bochum.de/share/Bericht_14-01.pdf

[Sc14] Schmidt, B.: Strategien für eine integrativ-nachhaltige Software-Entwicklung. Hochschule Bochum, Geoinformatik-Bericht Nr. 14-02, 2014. Abrufbar unter http://www.geoinformatik-bochum.de/share/Bericht_14-02.pdf

[SP12] Schweizer-Ries, P.; Perkins, D.: Sustainability Science: Transdisciplinarity, Transepistemology, and Action Research. Umweltpsychologie, 16(1), 2012; pp. 6-10.

[Ta11] Taina, J.: Good, Bad, and Beautiful Software : In Search of Green Software Quality Factors. CEPIS Upgrade, XII(4), Oct. 2011; pp. 22–27.

[Wi14] Wikipedia, Artikel "Digitale Nachhaltigkeit", http://de.wikipedia.org/wiki/Digitale_Nachhaltigkeit (Aufruf April 2014).

[WF89] Winograd, T.; Flores, F.: Erkenntnis, Maschinen, Verstehen. Berlin: Rotbuch, 1989.

[Wo87] World Commission on Environment, eds.: Report of the World Commission on Environment and Development: Our Common Future. Oslo: UN, 1987. Abrufbar als PDF-Dokument unter http://www.un-documents.net/wced-ocf.htm (Aufruf April 2014).

[Wy13] Wytzisk, A.; Schmidt, B.; Nüst, D.: Echtzeitinformation und Kollaboration in Geodateninfrastrukturen. zfv (Zeitschrift für Geodäsie, Geoinformation und Landmanagement), Heft 5/2013; S. 333-338.