www.oeko.de

Kriterienkatalog für nachhaltige Software

Lorenz Hilty, Stefan Naumann,
Yuliyan Maksimov, Eva Kern,
Andreas Filler, Achim Guldner
Jens Gröger

Version 01
Stand: 31.05.2017

UFOPLAN-Projekt „Sustainable Software Design - Entwicklung einer Methodik zur Bewertung der Ressourceneffizienz von Softwareprodukten“
FKZ-Nr.: 3715 37 601 0
Im Auftrag des Umweltbundesamtes

Projekt-Ansprechpartner

Dipl.-Ing. Jens Gröger
Öko-Institut e.V.
Tel. +49 30 40 50 85 378, j.groeger@oeko.de

Prof. Dr. Stefan Naumann
Hochschule Trier, Standort Umwelt-Campus Birkenfeld
Tel. +49 6782 17 12 17, s.naumann@umwelt-campus.de

Prof. Dr. Lorenz Hilty
University of Zurich
Tel. +41 44 635 67 24, hilty@ifi.uzh.ch

Inhaltsverzeichnis

Einleitung

Dieses Dokument ist Ergebnis des ersten von fünf Arbeitspaketen im UFOPLAN-Projekt „Sustainable Software Design – Entwicklung und Anwendung von Bewertungsgrundlagen für ressourceneffiziente Softwareprodukte unter Berücksichtigung bestehender Methodik“ des Umweltbundesamtes.

Ziel des Projekts ist es, eine Methodik zur Bewertung der Umweltwirkungen von Softwareprodukten zu entwickeln. Diese Methodik soll sowohl die Beschaffung von Softwareprodukten unter Berücksichtigung von Umweltkriterien als auch die Entwicklung ressourceneffizienter Software unterstützen. Insbesondere soll es möglich sein, zwei gegebene Softwareprodukte mit ähnlicher Funktionalität hinsichtlich ihrer Wirkungen auf natürliche Ressourcen zu vergleichen. Durch die Formulierung ambitionierter Mindeststandards sollen darüber hinaus Kriterien für die Kennzeichnung nachhaltiger Softwareprodukte durch ein Umwelt- oder Gütezeichen abgeleitet werden.

Das Projekt leistet damit einen Beitrag, den Fokus von „Green IT“ von der Hardware- auf die Softwareebene auszuweiten. Da Softwareprodukte immaterielle Güter sind, stellt sich dabei insbesondere das Problem, die indirekten materiellen Wirkungen dieser Produkte begrifflich und methodisch klar zu fassen.

Umweltwirkungen eines Produkts entstehen generell durch die Beanspruchung natürlicher Ressourcen[1] im Lebenszyklus des Produkts. Wir nehmen diese Lebenszyklusperspektive auch in Bezug auf Softwareprodukte ein (siehe Abbildung 0-1). Dabei berücksichtigen wir insbesondere, dass die für den Betrieb eines Softwareprodukts benötigte Hardware produziert, mit elektrischer Energie versorgt und am Ende ihrer Nutzungsdauer entsorgt werden muss. Jedes Softwareprodukt beansprucht somit einen quantifizierbaren Anteil am Lebenszyklus aller zu seinem Betrieb nötigen Hardwareprodukte (programmierbare Geräte in jeglicher Form, Peripheriegeräte und Datenträger).

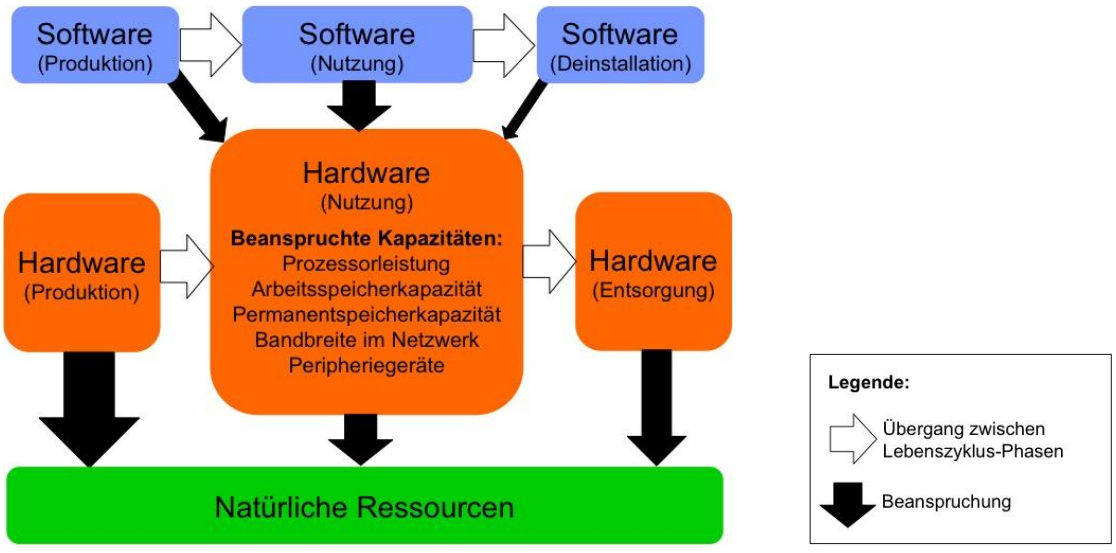


Abbildung 0-1 Lebenszyklen von Hardware und Software (horizontale Dimension) und Beanspruchung von Ressourcen (vertikale Dimension).

Aufgrund der Lebenszyklusperspektive ist dieser Ansatz geeignet für eine Erweiterung in Richtung auch der sozialen Aspekte der Rohstoffgewinnung und der Arbeitsbedingungen in der Hardwareproduktion oder -entsorgung; unser Fokus liegt jedoch auf den Umweltaspekten.

[1] Definitionen von „Ressource“ und anderen zentralen Begriffen sind im Glossar zu finden (Anhang C). Wir reservieren den Begriff der Ressource in diesem Dokument für natürliche Ressourcen und vermeiden den technischen Begriff der Hardwareressource weitgehend, indem wir Hardwareressourcen direkt durch ihre quantifizierbaren Leistungsdimensionen, also Kapazitäten, beschreiben.

Auf Softwareebene beschränken wir die Perspektive im Folgenden bewusst auf die Nutzungsphase. Ziel der hier definierten Kriterien ist es, ein Softwareprodukt aufgrund von Eigenschaften, zu bewerten, die in der Nutzungsphase beobachtbar sind, sei es durch die Nutzenden selbst oder durch Personen, die spezielle Tests durchführen. Eine zusätzliche Berücksichtigung der Produktionsphase der Software wäre aber durch Erweiterung des Ansatzes prinzipiell möglich. Wichtiger als eine *Bewertung* des Softwareproduktionsprozesses erscheint uns jedoch dessen *Beeinflussung,* unter anderem in Form von Empfehlungen an die Verantwortlichen für die Softwareentwicklung. Hierfür wird in einer späteren Arbeitsphase dieses Projekts ein Leitfaden entwickelt.

Generell betrachten wir in diesem Projekt Standard-Anwendungssoftware, also keine Systemsoftware und keine spezielle Anwendungssoftware für wenige Nutzende. Im letzteren Fall müsste der Ressourcenaufwand für die Entwicklung sicher miteinbezogen werden.

Gerade bei der Bewertung weit verbreiteter Softwareprodukte ist grundsätzlich nicht nur eine Momentaufnahme, sondern die Nutzung des Softwareprodukts über längere Zeiträume (auch mehrerer Versionen) zu betrachten. Erst aus dieser Perspektive wird z. B. die Frage der durch Software induzierten Beschaffung von Hardware relevant.

Abstrakt formuliert, stehen zwei durch ein Softwareprodukt verursachte Flüsse im Vordergrund unserer Betrachtung:

- der Durchfluss von Hardware durch die nutzende Organisation (neue Hardware zu Abfall),
- der Durchfluss von Energie durch die Hardware (Elektrizität zu Abwärme).

Wenn ein Softwareprodukt im Vergleich zu Konkurrenzprodukten mit ähnlicher Funktionalität einen deutlich geringeren Hardware- und Energiefluss verursacht, kann es als „nachhaltig“ gelten.[2]

Der Fluss von Hardware kann mit Methoden der Ökobilanzierung von Produkten (Life Cycle Assessment, LCA) hinsichtlich seiner Ressourcenbeanspruchung eingeschätzt werden. Hierfür gibt es Lebenszyklusinventare zur Produktion und Entsorgung der wichtigsten Hardwarekomponenten, die wir als gegeben voraussetzen und nicht weiter behandeln. Ebenso kann der Energiefluss mit Methoden der Ökobilanzierung bewertet werden; die verschiedenen Methoden der Stromerzeugung sind ausreichend untersucht, diese Daten setzen wir also auch als gegeben voraus.

Aus diesen Gründen ist es ausreichend, mit den hier entwickelten Kriterien den Einfluss von Software auf benötigte Hardwarekapazitäten zu adressieren. Stellt man sich eine Wirkungskette von Softwareeigenschaften bis hin zur Beanspruchung natürlicher Ressourcen vor, so behandeln wir hier also ausschließlich den von der Software bis zu Hardwareprodukten und ihrem Stromverbrauch[3] reichenden Abschnitt der Wirkungskette, denn nur dieser ist für unseren Untersuchungsgegenstand spezifisch.

Um Software hinsichtlich ihrer Nachhaltigkeit in Bezug auf den ausgelösten Hardware- und Energiefluss zu beurteilen, sind operationalisierbare Kriterien notwendig. Diese Kriterien können dann z. B. zur Information der Verantwortlichen für Softwareentwicklung, Softwarebeschaffung oder zur Vergabe eines Umweltkennzeichens eingesetzt werden.

---

[2] Die Funktionalität eines Softwareprodukts und damit sein *Nutzen* wird hier nicht bewertet. Ziel ist es ausschließlich, den ausgelösten *Aufwand* (an Ressourcen) abzuschätzen und zu bewerten. Bei gegebenem Nutzen lässt sich dieser zum Aufwand ins Verhältnis setzen, um die *Effizienz* zu ermitteln.

[3] In einigen Fällen kann eine Erweiterung dieser Perspektive notwendig sein, indem analog zum Energiefluss auch Flüsse von Verbrauchsmaterialien wie Papier oder Toner durch die Hardware relevant sind. Ob ein solcher Fall vorliegt, kann nach einem ersten Screening vor Anwendung der Kriterien entschieden werden.

Der hier vorgeschlagene Kriterienkatalog konzentriert sich auf Umweltwirkungen, die aus dem *Betrieb* eines Softwareprodukts resultieren. Dies schließt nicht aus, dass für die Vergabe eines Umweltkennzeichens weitere Kriterien zur Anwendung kommen, die sich auf den Prozess der Softwareentwicklung (z. B. Einhaltung von ILO[4]-Standards beim Outsourcing von Programmierarbeiten), auf die Funktionalität der Software (z. B. Barrierefreiheit oder Ausschluss bestimmter Kategorien wie „Ballerspiele“) oder weitere Aspekte beziehen. Es erscheint uns jedoch wichtig, die hier behandelten *Auswirkungen von Softwareeigenschaften auf die Beanspruchung natürlicher Ressourcen* als klar definierten Forschungsgegenstand zu behandeln und nicht schon im Ansatz mit anderen Fragestellungen zu vermischen. Häufig liegen für diese angrenzenden Fragestellungen auch schon Studien und Kriterien vor, die als Ergänzung unseres Kriterienkatalogs benutzt werden können.

Auf den folgenden Seiten ist ein Baum (eine Hierarchie) von Kriterien und Indikatoren beschrieben. Die Blätter des Baumes sind Indikatoren, die zur Operationalisierung des jeweils übergeordneten Kriteriums dienen. Einen Gesamtüberblick über die Kriterien gibt das Inhaltsverzeichnis dieses Dokuments.

Die Aggregation (d. h. die arithmetische oder logische Zusammenfassung) der Indikatoren zu einem Kriterium und danach der Kriterien zu Oberkriterien wird später durch ein Bewertungsmodell geschehen, das wir im Rahmen dieses Projekts der Struktur nach anlegen. Die eigentliche Bewertung durch die Zuweisung von Gewichten, Normierungsfunktionen oder die Ausweisung von MUSS-Kriterien bleibt dem Umweltbundesamt oder anderen Anwendern unserer Kriterien überlassen und kann sich über die Zeit ändern.

Die Gewichtung von Indikatoren und Kriterien kann auch in einem bestehenden Bewertungsmodell nach Softwareklassen variieren. Insbesondere können Indikatoren oder Kriterien im Grenzfall mit 0 gewichtet werden, wenn sie für bestimmte Softwareklassen nicht anwendbar oder irrelevant sind. Wir haben eine auf die Anwendung unserer Kriterien abgestimmte Klassifikation von Softwareprodukten entwickelt (siehe auch Anhang A):

- lokale Anwendung
- Anwendung mit entfernter Datenhaltung
- Anwendung mit entfernter Verarbeitung
- Serverdienst

Diese vier Klassen sind insofern von Bedeutung, als dass unser Ansatz nicht nur den Aufwand für die lokale Ausführung der Software, sondern auch den „remote“ ausgelösten Aufwand, von der Netzwerk-Infrastruktur über dedizierte Server bis hin zur Cloud, zu fassen versucht. Anderenfalls wäre der Ansatz nutzlos, weil die Kriterien durch Verlagerung von Umweltwirkungen an einen anderen Ort erfüllt werden könnten. Ein detailliertes Wirkungsmodell, das die verschiedenen Wirkungspfade von Softwareeigenschaften über Hardwarekapazitäten bis zu den natürlichen Ressourcen beschreibt, ist im Anhang B zu finden.

Im folgenden Hauptteil dieses Dokuments ist jedes Kriterium charakterisiert durch

- eine Nummer, je nach Ebene ein-bis dreistellig,
- eine Bezeichnung (Überschrift),
- eine Frage, die das Kriterium erläutert,
- evtl. einen auf die Frage folgenden Kommentar.

Das jeweils unterste Kriterium in der Hierarchie wird durch Indikatoren operationalisiert, die mit einem Kleinbuchstaben gekennzeichnet sind.

[4] International Labour Organization

Dieser Baum von Kriterien und Indikatoren basiert auf unserem Wirkungsmodell von Software (Anhang B) und einer umfassenden Literaturrecherche zu Kriterien zur Bewertung von Software, bei der mehr als 130 Kriterien zur Bewertung von Software aus über 60 Quellen analysiert wurden. Ein Entwurf des Kriterienkatalogs wurde auf einem Stakeholder-Workshop am 11. März 2016 in Berlin mit Fachpersonen aus Wissenschaft, Behörden und Industrie diskutiert. Das Feedback der Teilnehmenden während des Workshops und danach wurde bei der Überarbeitung des Dokuments berücksichtigt.

Einige der folgenden Kriterien und Indikatoren verweisen auf ein „Referenzsystem“, eine „Standardkonfiguration“ oder ein „Standardnutzungsszenario“; diese und weitere zentrale Begriffe sind in einem Glossar definiert (Anhang C).

1 Ressourceneffizienz

In welchem Ausmaß werden bei gegebener Funktionserfüllung Hardwarekapazitäten und damit indirekt natürliche Ressourcen beansprucht?

Dieses Hauptkriterium geht von der Vorstellung aus, dass die Erfüllung einer gegebenen Funktionalität durch ein Softwareprodukt mit unterschiedlicher Beanspruchung von Hardwarekapazitäten erfolgen kann, was indirekt zu einem unterschiedlichen Ausmaß in der Beanspruchung von natürlichen Ressourcen führt, die für die Bereitstellung und den Betrieb der Hardware benötigt werden.

Das angestrebte Ideal ist ein Softwareprodukt, das eine gegebene Funktionalität mit einem minimalen Ressourcenaufwand erbringt, also die Ressourceneffizienz (siehe Glossar) maximiert. Die Funktionalität wird durch Standardnutzungsszenarien (siehe Glossar) definiert. Als Näherung zur Abschätzung der beanspruchten natürlichen Ressourcen dienen die bereitzuhaltenden und die tatsächlich genutzten Hardwarekapazitäten sowie die dabei verbrauchte Energie.

1.1 Hardwareeffizienz

Welche Hardwarekapazitäten müssen für den Betrieb des Softwareprodukts bereitgehalten werden und wie werden sie während des Betriebs ausgelastet?

Die Hardwarekapazitäten werden in Prozent der entsprechenden Kapazität eines Referenzsystems[5] gemessen. Sie können nach zwei Dimensionen differenziert werden (**Tabelle 1-1**). Auf der einen Dimension werden sie nach lokalen, Netzwerk-, und entfernten Ressourcen differenziert. Hier unterscheiden wir zusätzlich zwischen den empfohlenen (1.1.1), minimalen (1.1.2), im Leerlauf benötigten (1.1.3) und bei einem Standardnutzungsszenario benötigten (1.1.4) Kapazitäten. Auf der anderen Dimension differenzieren wir nach dem Typ der Hardware-Kapazität: Rechenleistung, Arbeitsspeicher, Permanentspeicher, Bandbreite, Displayauflösung. Die Matrix ist offen für die Erweiterung um zusätzliche Spalten, wenn in Zukunft neue Kategorien von Hardware relevant werden sollten.

Tabelle 1-1 Differenzierung der Hardwarekapazitäten in zwei Dimensionen

		Rechenleistung	Arbeitsspeicher	Permanentspeicher	Bandbreite	Displayauflösung
lokal	empfohlen minimal Leerlauf Standardnutzung	1.1.1 a) 1.1.2 a) 1.1.3 a) 1.1.4 a)	1.1.1 b) 1.1.2 b) 1.1.3 b) 1.1.4 b)	1.1.1 c) 1.1.2 c) 1.1.3 c) 1.1.4 c)	-	1.1.1 d) 1.1.2 d)
Netz	empfohlen minimal Leerlauf Standardnutzung	-	-	-	1.1.1 e) 1.1.2 e) 1.1.3 d) 1.1.4 d)	-
entfernt	empfohlen minimal Leerlauf Standardnutzung	1.1.1 f) 1.1.2 f) 1.1.3 e) 1.1.4 e)	1.1.1 g) 1.1.2 g) 1.1.3 f) 1.1.4 f)	1.1.1 h) 1.1.2 h) 1.1.3 g) 1.1.4 g)	-	-

[5] Zur Anwendung des Kriteriensystems muss periodisch ein Referenzsystem auf dem Stand der technischen Entwicklung bestimmt werden. Dieses dient zur Normierung von Indikatoren.

Jede Zelle der in **Tabelle 1-1** dargestellten Matrix zeigt das zugehörige Kriterium (z. B. 1.1.1) mit dem zugehörigen Indikator (z. B. a)) zur Operationalisierung des Kriteriums. Die Kriterien und Indikatoren werden in den folgenden Abschnitten erläutert, die entsprechend nummeriert sind. Nicht alle Kriterien 1.1.1 – 1.1.4 sind in allen Feldern der Matrix anwendbar, deshalb bleiben einige Zellen leer.

Die Kriterien 1.1.5 und 1.1.6 dienen ebenfalls zur Beurteilung der Hardwareeffizienz, können pauschal beurteilt werden und erfordern keine Differenzierung nach dieser Matrix. Sie sind deshalb in **Tabelle 1-1** nicht erwähnt.

Grundsätzlich stellt sich für die spätere Aggregation von Kriterien das Problem einer Konkurrenz zwischen verschiedenen Hardwarekapazitäten (lokal vs. entfernt, Rechenleistung vs. Arbeitsspeicher, Rechenleistung zur Datenkompression vs. Bandbreite usw.). Könnte man die Hardwarekapazitäten in Form eines ökologischen Fußabdrucks bewerten, könnten sie mit dem Ergebnis dieser Bewertung gewichtet und aggregiert werden. Die Abschätzung des ökologischen Fußabdrucks ist nicht Teil der Arbeiten, die hier vorgestellt werden; wir verweisen den Leser auf existierende Lebenszyklus-Inventare von IKT-Hardware und elektrischer Energie als Grundlage für eine Aggregation.

Tabelle 1-2 Grundlegende Definitionen für die Messung der Kriterien 1.1.3 und 1.1.4

Bezeichner	Name	Definition	Anmerkung
VA_i	Vollauslastung	Oberen Grenze der Auslastung i des Referenzsystems	Bei Prozessorleistungen beträgt die Vollauslastung 100%, bei Arbeitsspeicher die Summe der installierten RAM-Kapazitäten, beim Netzwerk die maximale Übertragungsgeschwindigkeit usw.
GA_i	Grundauslastung	Messwert der mittleren Auslastung i des Referenzsystems ohne die zu untersuchende Software.	
LA_i	Leerlauf-Auslastung	Messwert der mittleren Auslastung i des Referenzsystems im Leerlauf der untersuchten Software	Sie beinhaltet sowohl die Grundauslastung (GA_i) als auch die zusätzliche durch den Leerlaufbetrieb der Software verursachte Hardware-Auslastung.
NLA_i	Netto-Leerlauf-Auslastung	$NLA_i = LA_i - GA_i$	Sie beschreibt die durch den Leerlaufbetrieb der Software über die Grundauslastung hinausgehende beanspruchte Hardware-Auslastung
t	Ausführungsdauer	Laufzeit der untersuchten Software zur Ausführung des Standardnutzungsszenarios auf dem Referenzsystem	Sie beginnt mit dem Start des Standardnutzungsszenarios und endet mit der Erledigung aller dort vorgesehenen Aktionen, inklusive nachgelagerter Prozesse

			(z.B. Freigeben von Arbeitsspeicher, Löschung temporärer Dateien).
BA_i	Brutto-Auslastung	Mittlere Auslastung des Referenzsystems bei Ausführung eines Standardnutzungsszenarios über dessen Ausführungsdauer	Sie wird als zeitgewichteter Mittelwert der Messwerte über die Ausführungsdauer berechnet.
NA_i	Netto-Auslastung	$NA_i = BA_i – GA_i$	Sie bezeichnet die durch die Software verursachte mittlere Hardware-Auslastung nach Abzug der Grundauslastung des Referenzsystems.
AF_i	Allokationsfaktor	$AF_i = NA_i / (VA_i - GA_i)$ (Allokationsfaktor für die Ausführung des Standardnutzungsszenarios)	Verhältnis der durch die Software beanspruchten Netto-Auslastung zu der maximal zur Verfügung stehenden Auslastung. Dabei ist berücksichtigt, dass maximal nur die Differenz zwischen der Vollauslastung (VA) und der Grundauslastung (GA) zur Verfügung steht.
AFL_i	Leerlauf-Allokationsfaktor	$AFL_i = NLA_i / (VA_i - GA_i)$ (Allokationsfaktor für den Leerlauf der Software)	
EA_i	Effektive Auslastung	$EA_i = NA_i + AF_i * GA_i$	
ELA_i	Effektive Leerlauf-Auslastung	$ELA_i = NLA_i + AFL_i * GA_i$	Genutzt um die Indikatoren zur Berechnung der Hardwareansprüche von Kriterium 1.1.3 zu berechnen
H_i	Hardware-Inanspruchnahme	$H_i = EA_i * t$	Genutzt um die Indikatoren zur Berechnung der Hardwareansprüche von Kriterium 1.1.4 zu berechnen

Die nachfolgende Abbildung stellt den Messzyklus beispielhaft dar und illustriert die verschiedenen Auslastungen und deren Bezeichnung.

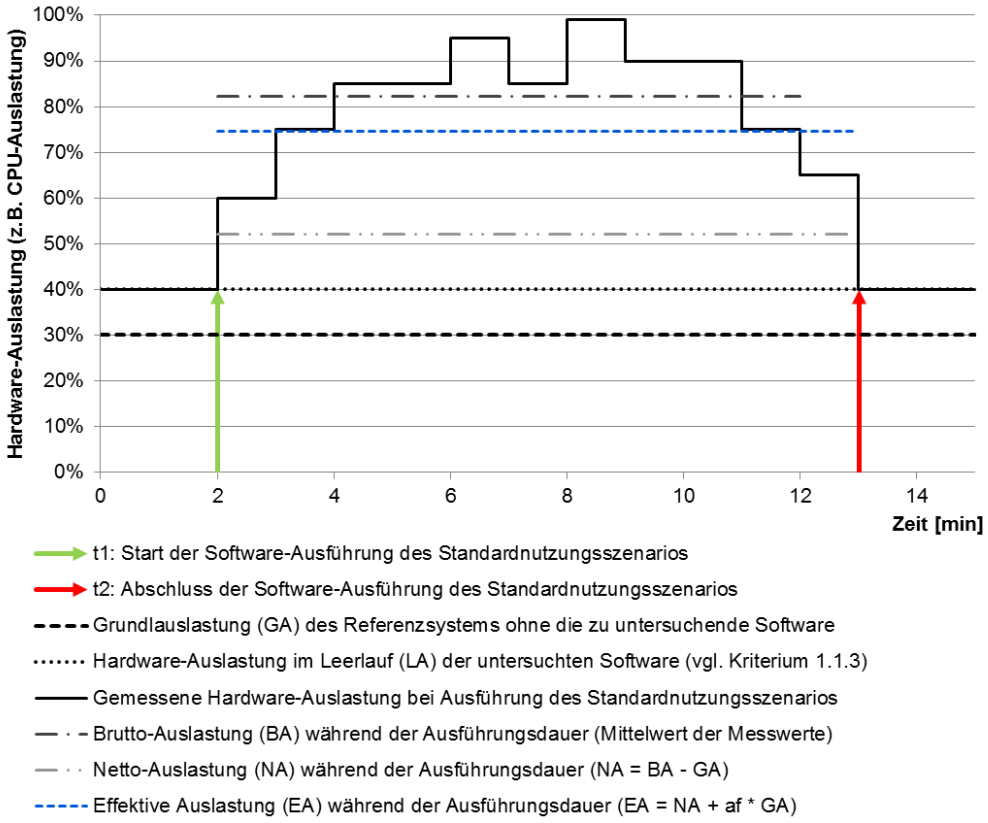


Abbildung 1-1 Beispielhafter Messzyklus zur Bestimmung der Hardware-Auslastung

1.1.1 Empfohlene Systemvoraussetzungen und resultierende Hardwareanforderungen (inkl. Peripheriegeräte)

Welche Systemvoraussetzungen für den Betrieb des Softwareprodukts werden vom Hersteller empfohlen? Wenn andere Softwareprodukte zur Nutzung des betrachteten Produktes vorausgesetzt werden, sind deren empfohlene Systemvoraussetzungen zusätzlich zu berücksichtigen.

Indikatoren:

a) Empfohlene lokale Rechenleistung laut Herstellerangaben in % der Rechenleistung des Referenzsystems (Produkt aus Taktfrequenz, Anzahl Kerne, Busbreite)
b) Empfohlener lokaler Arbeitsspeicher laut Herstellerangaben in % des Arbeitsspeichers des Referenzsystems
c) Empfohlener lokaler Permanentspeicher laut Herstellerangaben in % des Permanentspeichers des Referenzsystems
d) Empfohlene Displayauflösung laut Herstellerangaben in % der Displayauflösung des Referenzsystems
e) Empfohlene Bandbreite für Netzzugang laut Herstellerangaben in % der Bandbreite des Referenzsystems
f) Empfohlene serverseitige Rechenleistung laut Herstellerangaben in % der Rechenleistung des Referenzsystems

g) Empfohlener serverseitiger Arbeitsspeicher laut Herstellerangaben in % des serverseitigen Arbeitsspeichers des Referenzsystems
h) Empfohlener serverseitiger Permanentspeicher laut Herstellerangaben in % des serverseitigen Permanentspeichers des Referenzsystems

1.1.2 *Minimale* Systemvoraussetzungen und resultierende Hardwareanforderungen (inkl. Peripheriegeräte)

Welche Systemvoraussetzungen sind für den Betrieb des Softwareprodukts minimal notwendig?

Indikatoren:

a) Minimale lokale Rechenleistung laut Herstellerangaben in % der Rechenleistung des Referenzsystems
b) Minimaler lokaler Arbeitsspeicher laut Herstellerangaben in % des Arbeitsspeichers des Referenzsystems
c) Minimaler lokaler Permanentspeicher laut Herstellerangaben in % des Permanentspeichers des Referenzsystems
d) Minimale Displayauflösung laut Herstellerangaben in % der Displayauflösung des Referenzsystems
e) Minimale Bandbreite für Netzzugang laut Herstellerangaben in % der Bandbreite des Referenzsystems
f) Minimale serverseitige Rechenleistung laut Herstellerangaben in % der Rechenleistung des Referenzsystems
g) Minimaler serverseitiger Arbeitsspeicher laut Herstellerangaben in % des serverseitigen Arbeitsspeichers des Referenzsystems
h) Minimaler serverseitiger Permanentspeicher laut Herstellerangaben in % des serverseitigen Permanentspeichers des Referenzsystems

1.1.3 Hardware-Auslastung im Leerlauf unter der Annahme einer Standardkonfiguration

Wie hoch ist die mittlere Auslastung der bereitgestellten Hardwarekapazitäten durch das Softwareprodukt, wenn sich dieses im Leerlauf befindet?

Indikatoren:

a) Messung der mittleren Prozessorauslastung im Leerlauf unter Standardkonfiguration (Differenz zum Grundverbrauch der Standardkonfiguration ohne das Softwareprodukt im gleichen Zeitraum)
b) Messung der mittleren Arbeitsspeicherbelegung im Leerlauf unter Standardkonfiguration
c) Messung der mittleren Permanentspeicherbelegung im Leerlauf unter Standardkonfiguration
d) Messung der mittleren beanspruchten Bandbreite für Netzzugang im Leerlauf unter Standardkonfiguration
e) Messung der mittleren serverseitigen Prozessorauslastung im Leerlauf unter Standardkonfiguration
f) Messung der mittleren serverseitigen Arbeitsspeicherbelegung im Leerlauf unter Standardkonfiguration
g) Messung der mittleren serverseitigen Permanentspeicherbelegung im Leerlauf unter Standardkonfiguration

Die mittleren Prozessorauslastungen (Indikatoren a) und e)) sowie die mittleren Arbeitsspeicherbelegungen (Indikatoren b) und f)) werden als Effektive Leerlauf-Auslastungen (ELA) berechnet (siehe **Tabelle 1-2**). Sie beinhalten neben der Netto-Leerlauf-Auslastung (NLA) zusätzlich noch einen Anteil an der Grundauslastung (GA) gemäß den Festlegungen in Abschnitt 1.1.

1.1.4 Hardware-Inanspruchnahme bei normaler Nutzung unter der Annahme einer Standardkonfiguration und eines Standardnutzungsszenarios

Wie hoch ist die Inanspruchnahme der bereitgestellten Hardwarekapazitäten beim Betrieb des Softwareprodukts?

Als Hardware-Inanspruchnahme wird hier das Integral der Hardware-Auslastung über die Ausführungsdauer eines Standardnutzungsszenarios verstanden. Die Maßeinheiten für die Hardware-Inanspruchnahme sind Einheiten für Arbeitsleistung, wie %*s (Prozessorarbeit), MByte*s (Arbeitsspeicherarbeit) und MBit/s*s = MBit (im Netzwerk übertragene Datenmenge). Anders als bei den vorangehenden Kriterien 1.1.1 – 1.1.3 wird bei der Hardware-Inanspruchnahme also auch die Ausführungsdauer berücksichtigt. Zur Erläuterung: Wenn ein Programm A doppelt so viel Prozessorleistung, Arbeitsspeicher oder Bandbreite beansprucht wie Programm B, um ein gegebenes Standardnutzungsszenario zu erledigen, aber dafür den Prozessor, Speicher oder die Bandbreite nach der Hälfte der von B benötigten Zeit wieder freigibt, so ist die Hardware-Inanspruchnahme beider Programme gleich hoch.

Die Hardware-Inanspruchnahme wird als Produkt aus mittlerer effektiver Hardware-Auslastung und der zur Ausführung des Standardnutzungsszenarios benötigten Zeit berechnet. Die Ausführungsdauer ist dabei für alle Hardwarekapazitäten gleich hoch.

Indikatoren:

a) Messung der Prozessorarbeit bei Ausführung des Standardnutzungsszenarios unter Standardkonfiguration
b) Messung der Arbeitsspeicherarbeit bei Ausführung des Standardnutzungsszenarios unter Standardkonfiguration
c) Messung der Permanentspeicherarbeit bei Ausführung des Standardnutzungsszenarios unter Standardkonfiguration
d) Messung der übertragenen Datenmenge für Netzzugang bei Ausführung des Standardnutzungsszenarios unter Standardkonfiguration
e) Messung der serverseitigen Prozessorarbeit bei Ausführung des Standardnutzungsszenarios unter Standardkonfiguration
f) Messung der serverseitigen Arbeitsspeicherarbeit bei Ausführung des Standardnutzungsszenarios unter Standardkonfiguration
g) Messung der serverseitigen Permanentspeicherarbeit bei Ausführung des Standardnutzungsszenarios unter Standardkonfiguration

Die Hardware-Inanspruchnahme (H) für die Prozessorarbeit (Indikatoren a) und e)) sowie die Arbeitsspeicherarbeit (Indikatoren b) und f)) werden berechnet wie in **Tabelle 1-2** angegeben.

1.1.5 Sparsame Hardwarenutzung durch Anpassbarkeit und Unterstützung der Nutzenden bei der Anpassung des Softwareprodukts

Beansprucht das Softwareprodukt nur jene Hardwarekapazitäten, die für die Ausführung der von den jeweiligen Nutzenden nachgefragten Funktionen benötigt werden? Werden die Nutzenden durch die Software ausreichend dabei unterstützt, entsprechende Anpassungen vorzunehmen?[6]

Indikatoren:

a) Geschieht die Minimierung der beanspruchten Kapazitäten automatisch und/oder gibt es bei der Installation der Software entsprechende Optionen? (Skala: ja/nein)
b) Falls die Nutzenden eine entsprechende Wahl treffen, ist die Entscheidung für oder gegen Installationsoptionen später jederzeit revidierbar (Skala: ja/nein)
c) Black-Box-Test: Können hardwareintensive Software-Module abgeschaltet werden? (Skala: können abgeschaltet werden/können nicht abgeschaltet werden)
d) Ist es ohne Nachteil möglich, vorübergehend oder dauerhaft nicht benötigte Peripheriegeräte abzuschalten bzw. gar nicht bereitzustellen? (Skala: Können vorübergehend und dauerhaft abgeschaltet werden/können nur vorübergehend abgeschaltet werden/können nicht abgeschaltet werden)
e) Werden nach der Installation die Dateien gelöscht, die nur zur Installation benötigt werden?

1.1.6 Online-Auslieferung

Kann das Softwareprodukt (inkl. aller Programme, Daten und Dokumentation einschließlich Handbüchern) ohne den Transport physischer Datenträger (inkl. Papier) oder anderer materieller Güter (inkl. Verpackung) erworben, installiert und betrieben werden?

Indikatoren:

a) Sind eine Online-Auslieferung und ein Online-Update der Software möglich?
b) Wird unterstützt, dass das Softwareprodukt und seine Updates in der anwendenden Organisation zentral abgelegt werden?

1.2 Energieeffizienz

Wie viel elektrische Energie verbraucht die Hardware bei Nutzung des Softwareprodukts zur Ausführung eines Standardnutzungsszenarios?

Der Verbrauch elektrischer Energie ist eine Konsequenz der Beanspruchung von Hardwarekapazitäten. Deren Bestimmung wurde unter 1.1.4 bereits behandelt. Deshalb soll die elektrische Leistungsaufnahme der Hardware im Rahmen der unter 1.1.4 beschriebenen Messungen jeweils mitgemessen werden, zumindest für die Summe der jeweils lokal, im Netz oder entfernt genutzten Hardwarekapazitäten.

Indikatoren:

a) Messung der zur Ausführung des Standardnutzungsszenarios unter Standardkonfiguration auf dem lokalen Gerät verbrauchten Energie
b) Schätzung der durch die Datenübertragung im Netz verbrauchten Energie aufgrund des bei Ausführung des Standardnutzungsszenarios unter Standardkonfiguration erzeugten

[6] keine Belegung von Kapazitäten durch vorübergehend oder dauerhaft nicht nachgefragte Funktionalität.

Datenverkehrs (Verwendung einer aktuellen Schätzung für die Energieintensität des Netzes in kWh/GB basierend auf aktueller Fachliteratur, wenn nötig differenziert nach Zugangsnetzen)

c) Messung der durch die entfernte Speicherung und Verarbeitung in Servern verbrauchten Energie bei Ausführung des Standardnutzungsszenarios unter Standardkonfiguration (falls Messung nicht möglich, Schätzung mit Hilfe durchschnittlicher Faktoren für die Energieintensität von Rechenzentrumsdienstleistungen basierend auf aktueller Fachliteratur)

Die verbrauchte elektrische Energie ergibt sich durch Integration der elektrischen Leistungsaufnahme über die Ausführungszeit des Standardnutzungsszenarios. Abweichend von den Festlegungen, die für die Hardware-Auslastung (siehe Abschnitt 1.1.4) getroffen wurden, wird für die Messungen der elektrischen Energie (Indikatoren a) und c)) nur der Nettowert der elektrischen Leistungen ermittelt, also nur die Leistungswerte, die über den elektrischen Grundverbrauch des Referenzsystems hinausgehen. Diese abweichende Berechnung erfolgt sowohl aus Gründen der Praktikabilität (die Berechnung des Allokationsfaktors nach 1.1.4 ist bei elektrischen Leistungswerten aufgrund fehlender Obergrenzen nicht praktikabel) als auch aus Gründen der Aussagekraft (die Unterschiede im Energieverbrauch von Software werden deutlicher, wenn die Grundauslastung nicht im Indikator enthalten ist).

1.3 Ressourcenmanagement

In welchem Ausmaß trägt das Softwareprodukt während seines Betriebs zu einem effizienten Management der von ihm beanspruchten Ressourcen bei?

Weil das Ausmaß der Nutzung eines gegebenen Softwareprodukts schwanken kann, trägt eine entsprechende adaptive und durch das Softwareprodukt unterstützte Nachfrage nach Hardwarekapazitäten zur Ressourcenschonung bei. Freigegebene Hardwarekapazitäten können potenziell durch andere Prozesse genutzt werden oder ihren Energieverbrauch reduzieren. Beides trägt indirekt zur Schonung natürlicher Ressourcen bei.

Dieses Kriterium bezieht sich im Gegensatz zu den Kriterien 1.1. und 1.2 auf die Anpassung des Bedarfs an Hardwarekapazitäten zur Laufzeit des Programms, insbesondere den Übergang in sparsamere Modi, abhängig von den momentanen Anforderungen der Nutzenden oder dem Angebot an Hardwarekapazitäten und Energie. Während die Ressourceneffizienz in den verschiedenen Modi durch die Kriterien 1.1 und 1.2 adressiert wurde, steht hier also die Fähigkeit zum kontextabhängigen Wechsel zwischen den Modi im Vordergrund.

1.3.1 Anpassung der beanspruchten Kapazitäten an den Bedarf

Kann das Softwareprodukt im laufenden Betrieb Hardwarekapazitäten freigeben (und damit auch seinen Energieverbrauch reduzieren), wenn diese nicht benötigt werden?

Indikatoren:

a) Verfügt das Softwareprodukt über unterschiedliche Modi, deren Wechsel sich messbar auf den Energieverbrauch auswirkt?
b) Wechselt das Softwareprodukt dynamisch jeweils in einen sparsameren Modus, wenn das möglich ist? (z. B. Schlafmodus)
c) Falls für die Software spezielle Einstellungen für einen sparsamen Betriebsmodus vorgenommen werden müssen, sind diese Konfigurationsmöglichkeiten zentral und allgemein verständlich zusammengefasst?[7]

1.3.2 Anpassung des Bedarfs an die verfügbaren Kapazitäten

Kann das Softwareprodukt im laufenden Betrieb seinen Bedarf an Hardwarekapazitäten (und damit auch seinen Energieverbrauch) reduzieren, wenn das Angebot sich dynamisch verringert? (z. B. wenn weniger Bandbreite zur Verfügung steht oder der Akku sich leert)

Indikatoren:

a) Wechselt das Softwareprodukt in einen sparsameren Modus, wenn das Angebot an Hardwarekapazitäten oder Energie sich verringert, ohne dass Fehler oder Datenverluste auftreten? (keine Einschränkung, langsamere Ausführung, Fehler in der Ausführung)
b) Ist die Software auch bei aktiviertem Energiemanagement der darunterliegenden Systemschichten oder der verbundenen Clientsysteme uneingeschränkt funktional nutzbar?[8] (ja, uneingeschränkt nutzbar / eingeschränkt nutzbar / nicht nutzbar)

1.3.3 Ressourcenschonende Standardeinstellungen

Sind die Standardeinstellungen des Softwareprodukts so gewählt, dass sie das Ziel der Ressourcenschonung mitberücksichtigen?[9]

Indikatoren:

a) Einschätzung des Prüfers, ob Standardeinstellungen des Softwareprodukts so gewählt sind, dass sie das Ziel der Ressourcenschonung mitberücksichtigen

---

[7] Beispiele: Hintergrund-/Sleep-Einstellungen, Animationen, rechenintensive Prozesse wie Index etc., Cache-Größen, zeitliche Steuerbarkeit von Prozessen, um ökologisch günstigere Energiezufuhr auszunutzen (demand shaping).

[8] Insbesondere ist bei serverbasierter Software zu fordern, dass das Aktivieren des Energiemanagements auf Client-Seite die Funktionalität nicht beeinträchtigt. Beispielsweise darf es nicht zu einem Verlust von Session-Information führen, wenn der Client in einen Sleep-Mode wechselt.

[9] Beispiel: Standardeinstellung beim Drucken: Wird Papier beidseitig bedruckt, wenn der Drucker dazu in der Lage ist?

1.3.4 Feedback zur Beanspruchung von Hardwarekapazitäten und Energie

Können die vom Softwareprodukt aktuell beanspruchten lokalen und entfernten Hardwarekapazitäten und resultierenden Energieverbräuche angezeigt werden und ist die Anzeige korrekt?

Indikatoren:

a) Werden die beanspruchten Hardwarekapazitäten, der Datenfluss und die Energieverbräuche angezeigt? (funktionsspezifisch vorhanden mit Vorschlägen zur Ressourceneinsparung, funktionsspezifisch vorhanden, für das Gesamtprodukt vorhanden, nicht vorhanden)
b) Einschätzung des Prüfers, ob Anzeige korrekt (korrekt / nicht korrekt)

2 Potentielle Hardware-Nutzungsdauer

Zu welchem Grad sind Hardware-Erneuerungszyklen von Software-Erneuerungszyklen entkoppelt?[10]

Software stellt Anforderungen an die Hardware, auf der sie ausgeführt wird. Je schneller diese Anforderungen mit der Weiterentwicklung des Softwareprodukts wachsen und je spezifischer sie sind, desto einschränkender wirken sie sich auf den Einsatz bereits vorhandener Hardwareprodukte aus. Können bereits vorhandene Hardwareprodukte nicht (mehr) eingesetzt werden, um das gegebene Softwareprodukt auszuführen, verkürzt dies die Nutzungsdauer der Hardware.

Das angestrebte Ideal ist ein Softwareprodukt, dessen Entwicklungsdynamik dem Betreiber ein davon unabhängiges, entkoppeltes Management seiner Hardwareprodukte erlaubt.

2.1 Abwärtskompatibilität

Garantiert der Hersteller des Softwareprodukts, dass das aktuelle Release auf einem Referenzsystem von vor *n* Jahren betrieben werden kann?[11]

Indikatoren:

a) Zunächst Herstelleraussage (Hardware, ältere Betriebssysteme, ältere Frameworks), da für zurückliegende Jahre keine Standardkonfigurationen definiert sind
b) Wenn seit der ersten Anwendung dieses Kriteriums auf das Softwareprodukt ausreichend Zeit verstrichen ist, so dass man das Standardnutzungsszenario auch auf früheren Standardkonfigurationen ausführen kann: Ist das Standardnutzungsszenario mit dem aktuellen Release des Softwareprodukts auf einer Konfiguration ausführbar, die vor einer noch festzulegenden Anzahl von Jahren Standardkonfiguration war?

[10] Entkopplung der Software- und Hardware-Erneuerungszyklen bedeutet hohe potenzielle Hardware-Nutzungsdauer. Grundannahme: Jedes Softwareprodukt benötigt eine Systemumgebung als Plattform, auf der es ausgeführt wird. Alle zur Ausführung notwendigen Hard- und Softwarekomponenten des IKT-Systems zählen zu dieser Systemumgebung. Das Softwareprodukt selbst kann Teil der Systemumgebung anderer Softwareprodukte sein. Beispiel: Ein Webbrowser benötigt ein Betriebssystem, weitere Systemsoftware und Hardware als Systemumgebung und bildet zugleich die Systemumgebung für eine Web-Anwendung. Von einem gegebenen Softwareprodukt aus betrachtet interessiert es, welche Ansprüche seine Erneuerung über die verschiedenen Schichten der Systemumgebung an die unterste Schicht – die Hardware – auslöst.

[11] Somit kann das Softwareprodukt mit einer üblichen Hardwarekonfiguration betrieben werden, die schon *n* Jahre in Betrieb ist.

2.2 Plattformunabhängigkeit und Portabilität

Kann das Softwareprodukt auf verschiedenen aktuell verbreiteten produktiven Systemumgebungen (Hardware und Software) betrieben werden und können die Nutzenden zwischen diesen ohne Nachteil wechseln?[12]

Indikatoren:

a) Herstellerangaben (kompatibel mit verschiedenen Betriebssystemen, Laufzeitumgebungen)
b) Standardnutzungsszenario auf verschiedenen aktuell verbreiteten produktiven Systemumgebungen ausführen, dabei die Daten- und Softwareeinstellungsportabilität prüfen

2.3 Hardwaresuffizienz

Bleibt die Menge an beanspruchter Hardwarekapazität bei Weiterentwicklung des Softwareprodukts auch bei Funktionserweiterungen über die Zeit konstant?

Dieses Kriterium belohnt Hersteller von Softwareprodukten, die keine über die Zeit wachsenden Anforderungen an die Hardware stellt, wodurch die Hardware nicht aufgerüstet oder erneuert werden muss. Es berücksichtigt bewusst nicht, ob die Funktionalität erweitert wurde. Suffizienz ist so zu verstehen, dass die beanspruchten Ressourcen auch dann *nicht* zunehmen, wenn man mehr Nutzen aus ihnen zieht (was durch Zunahme der Effizienz möglich ist).

Das angestrebte Ideal ist ein Softwareprodukt, das von Version zu Version zwar höheren Anforderungen der Nutzenden genügt, aber dennoch keine höheren Anforderungen an die Hardware stellt.

Dieses Kriterium ist erst anwendbar, wenn Produkte schon mehrfach beurteilt wurden, also mindestens ein zurückliegendes Ergebnis schon vorliegt.

Indikatoren:

a) Intertemporale Vergleiche mit folgenden Ergebnisstufen:

1. „sehr gut“: Die Versionswechsel haben bisher zu einer Verringerung der benötigten Hardwarekapazitäten geführt.
2. „gut“: Die Versionswechsel haben bisher nicht zu einer Erhöhung der benötigten Hardwarekapazitäten geführt.
3. „genügend“: Die Versionswechsel haben bisher zwar zu einer Erhöhung der benötigten Hardwarekapazitäten geführt, diese blieben jedoch im Rahmen des technisch bedingten Effizienzfortschritts, wie er sich in der zeitlichen Abfolge von Referenzsystemen ausdrückt.
4. „ungenügend“: Die Versionswechsel haben dazu geführt, dass die benötigten Hardwarekapazitäten schneller gewachsen sind als die technische Effizienz entsprechend der Abfolge der Referenzsysteme.

[12] Wir empfehlen, dieses Kriterium nicht zu den Minimalanforderungen zu zählen, da es grundsätzlich sehr ressourcenschonende Software geben kann, die nur auf einer einzigen Plattform läuft. Dennoch ist Plattformunabhängigkeit positiv zu werten, da sie die Freiheitsgrade des Nutzenden bei der Optimierung der Beschaffung Hardware und Systemsoftware vergrößert.

3 Nutzungsautonomie

Respektiert der Hersteller des Softwareprodukts die Autonomie des Nutzenden im Umgang mit dem erworbenen Produkt?

Dieses Hauptkriterium geht von der Annahme aus, dass eine relevante Zahl von Nutzenden an einem ressourcenschonenden Einsatz von Software interessiert ist. Wenn es ihnen ohne funktionelle Nachteile möglich ist, werden sie versuchen, mit geringer Hardwarekapazität auszukommen (für die sie in der Regel bezahlen) und auch den Energieverbrauch gering zu halten (der ebenfalls finanziell relevant ist oder zumindest die Akkulaufzeit mobiler Geräte beeinflusst). Dies ist jedoch nur unter der Voraussetzung möglich, dass die Nutzenden nicht zu unnötiger Beanspruchung von Ressourcen gezwungen werden und verstehen, wie sie sie vermeiden können.

Das angestrebte Ideal ist ein Softwareprodukt, das die Freiheit der Nutzenden über die Beanspruchung von Kapazitäten (und damit indirekt von Ressourcen) bei Nutzung des Produkts selbst zu entscheiden, möglichst weitgehend respektiert.

Die folgenden Kriterien sind aus der Perspektive von technikfernen Zielgruppen zu beurteilen, sie sind also in der Regel nicht dadurch schon erfüllt, dass sie bei Nutzung des Produkts durch eine speziell ausgebildete oder speziell geübte Fachperson erfüllt wären. Eine Ausnahme hiervon bildet Kriterium 3.1.2.

3.1 Transparenz und Interoperabilität

Sind ressourcenrelevante Aspekte des Softwareprodukts für Nutzende mit vernünftigem Aufwand nachvollziehbar? Können Nutzende die Daten, die sie mit dem Softwareprodukt erzeugt haben, mit anderen Softwareprodukten weiterverwenden?

3.1.1 Transparenz der Datenformate und Datenportabilität

Sind die Datenformate (Datei- oder Datenstromformate), die das Softwareprodukt zur Ablage der von Nutzenden erzeugten Daten oder zum Austausch von Daten mit anderen Programmen verwendet, ausreichend dokumentiert, um Interoperabilität zu ermöglichen? Folgen die Datenformate offenen Standards, so dass eine Weiternutzung der Daten mit einem anderen Softwareprodukt möglich ist?[13]

Zur Anwendung dieses Kriteriums muss definiert sein, welche Standards zum Zeitpunkt der Vergabe zu den offenen Standards gezählt werden.

Indikatoren:

a) Überprüfung der Handbücher und technischen Datenblätter, ob Datenformate ausreichend dokumentiert sind
b) Abgleich mit bekannten und offenen Standards

[13] Dies ist entscheidend für Vermeidung einer Abhängigkeit vom Softwareprodukt (Customer Lock-In), welche unnötigen Ressourcenverbrauch erzwingen kann, sowohl im Falle der Beibehaltung eines ineffizienten Produkts als auch im Falle eines (aufwändigen) Umstiegs auf ein anderes Produkt.

3.1.2 Transparenz und Interoperabilität der Programme

Sind Anwendungs-Programmier-Schnittstellen (APIs) klar dokumentiert und wird die Verbreitung und Weiterentwicklung des Programms unterstützt? Folgen die Schnittstellen offenen Standards, die Interoperabilität ermöglichen?

Die Gewichtung der Indikatoren ist möglicherweise stark kontextabhängig. Die Auswirkung von offenem Quellcode und bestimmter Lizenzmodelle auf die Beanspruchung von Ressourcen kann nicht im Sinne einer generellen Regel beurteilt werden.

Indikatoren:

a) Wenn es APIs gibt: Überprüfung der Schnittstellendokumentation anhand der Dokumentation des Softwareprodukts und seiner APIs
a) Ist der Quellcode vollständig offengelegt?
b) Ist die Software unter einer Lizenz veröffentlicht, die es erlaubt, die Software weiterzuentwickeln?

3.1.3 Kontinuität des Softwareproduktes

Kann das Softwareprodukt über längere Zeiträume genutzt werden, ohne dass schwerwiegende Nachteile (insbesondere Probleme der IT-Sicherheit) auftreten, und hat der Nutzende die Wahl, unnötige Updates zu vermeiden?[14]

Indikatoren:

a) Wie lang ist der Zeitraum, für den der Anbieter die zukünftige Unterstützung des Produkts mit Sicherheitsupdates garantiert?
b) Reagiert der Hersteller zeitnah auf das Bekanntwerden von Sicherheitslücken?
c) Kann der Nutzende durch Konfiguration die Updatehäufigkeit beeinflussen und dabei insbesondere zwischen Sicherheitsupdates und sonstigen Updates differenzieren?
d) Besteht die Möglichkeit, nur differenzielle Updates zu erhalten?[15]

3.1.4 Transparenz des Prozessmanagements

Macht das Softwareprodukt die Nutzenden darauf aufmerksam, dass es im Hintergrund automatisch Prozesse startet oder weiterlaufen lässt, die möglicherweise nicht genutzt werden?

Indikatoren:

a) Prüfung anhand der Installation und der Ausführung von Standardnutzungsmustern, welche Prozesse das Softwareprodukt automatisch startet und ob es darauf aufmerksam macht (es macht auf alles aufmerksam/auf einiges/macht nicht aufmerksam)
b) Wenn das Softwareprodukt beim Systemstart automatisch gestartet wird („Autostart"): Macht es darauf aufmerksam, dass dies der Fall ist?

[14] Eine hohe Updatefrequenz verursacht Ressourcenverbrauch und erschwert die Aufrechterhaltung von Transparenz. Das Kriterium der Notwendigkeit von Updates ist grundsätzlich schwer objektivierbar; eine Unterscheidung zwischen sicherheitsrelevanten (und damit zweifellos notwendigen) und anderen Updates ist in jedem Fall sinnvoll und wird daher durch Indikator b) adressiert.

[15] Dies vermeidet den Austausch des kompletten Programms, der bei häufigen Updates erheblichen Ressourcenaufwand verursachen kann.

c) Wenn der Nutzende eine Aktion durchführt, die als Beendigung des Programms aufgefasst werden kann, aber mindestens einer der Prozesse noch aktiv bleibt: Macht das Softwareprodukt darauf aufmerksam?

3.2 Deinstallierbarkeit

Lässt sich das Softwareprodukt einfach, rückstandsfrei und ohne vermeidbare Nachteile deinstallieren?

3.2.1 Deinstallierbarkeit der Programme

Werden Nutzende ausreichend darin unterstützt, das Softwareprodukt rückstandsfrei zu deinstallieren?

Indikatoren:

a) Deinstallation der Software und Vergleich mit dem Zustand vor der Installation, der gleich sein muss.

3.2.2 Löschbarkeit der Daten

Werden Nutzende ausreichend darin unterstützt, die während des Betriebs des Softwareprodukts generierten Daten, die sie nicht explizit angelegt haben, nach Bedarf zu löschen?

Dieses Kriterium zielt darauf ab, dass die vorübergehende Nutzung eines Softwareprodukts keine Datenspuren hinterlässt, wenn die Nutzenden dies nicht wünschen. Wenn später beispielsweise eine andere Version des Produkts oder ein Konkurrenzprodukt installiert wird, soll die Historie für die neue Software nicht erkennbar sein. Damit soll Lock-In-Effekten und der Missachtung von Datenschutzprinzipien vorgebeugt werden.

Indikatoren:

a) Ist der Zustand nach dem Löschen der nicht durch den Nutzer ausdrücklich gespeicherten Daten mit dem Zustand vor der Installation in relevanter Hinsicht identisch?
b) Macht das Softwareprodukt transparent, wo die explizit angelegten Daten gespeichert werden (Speicherort)?
c) Wird der Nutzende dabei unterstützt, die auf entfernten Speicherkapazitäten angelegten Datenbestände zu löschen?

3.3 Wartungsfunktionen

Bietet das Softwareprodukt einfach zu benutzende Funktionen, die es erlauben, eingetretene Schäden an Daten und Programmen zu beheben?

3.3.1 Datenwiederherstellbarkeit

Lässt sich der letzte Zustand der Daten nach einem unerwünschten Programmabbruch wiederherstellen?

Indikatoren:

a) Macht der Hersteller dazu Angaben, und lassen sich diese im Test bestätigen?
b) Ist der Zeitraum, in dem Änderungen an den Daten zwischengespeichert werden, parametrisierbar?

3.3.2 Selbstreparaturfähigkeit

Lässt sich die installierte Instanz des Softwareprodukts nach dem Eintreten eines inkonsistenten Zustands wiederherstellen?

Indikatoren:

a) Herstellerangaben und Überprüfung durch Test

3.4 Unabhängigkeit von Fremdressourcen

Lässt sich das Softwareprodukt möglichst unabhängig von Hardwarekapazitäten betreiben, die nicht der Kontrolle der Nutzenden unterliegen?

3.4.1 Offlinefähigkeit

Zu welchem Grad vermeidet das Softwareprodukt Zwänge zur Konnektivität, die nicht aus der Funktionserfüllung resultieren?[16]

Indikatoren:

a) Überprüfung durch Standardnutzungsszenario (Offline-Betrieb möglich/mit Einschränkungen möglich/unmöglich)

[16] Beispiele für unnötige Konnektivitätszwänge: Verbindung zum Lizenzserver herstellen, wiederholtes Herunterladen von Fonts notwendig.

3.5 Qualität der Produktinformation

Unterstützt die angebotene Information über das Softwareprodukt seine ressourcenschonende Nutzung?

3.5.1 Verständlichkeit und Überschaubarkeit der Produktdokumentation, Lizenz- und Nutzungsbedingungen

Sind alle Angaben für die Nutzenden leicht nachvollziehbar?

Indikatoren:

a) Augenscheinprüfung durch Prüfende; Test mit tatsächlichen Nutzenden

3.5.2 Ressourcenrelevanz der Produktinformation

Enthält die Produktinformation alle Angaben, die die Nutzenden benötigen, um die Ressourcenbeanspruchung durch das Softwareprodukt gering zu halten, in strukturierter Form, und sind die Angaben korrekt?

Langfristiges Ziel ist die Entwicklung standardisierter Produktbeschreibungen für ressourcenrelevante Produktinformation. Sobald es hierzu einen befriedigenden Standard gibt, kann seine Einhaltung als Indikator aufgenommen werden.

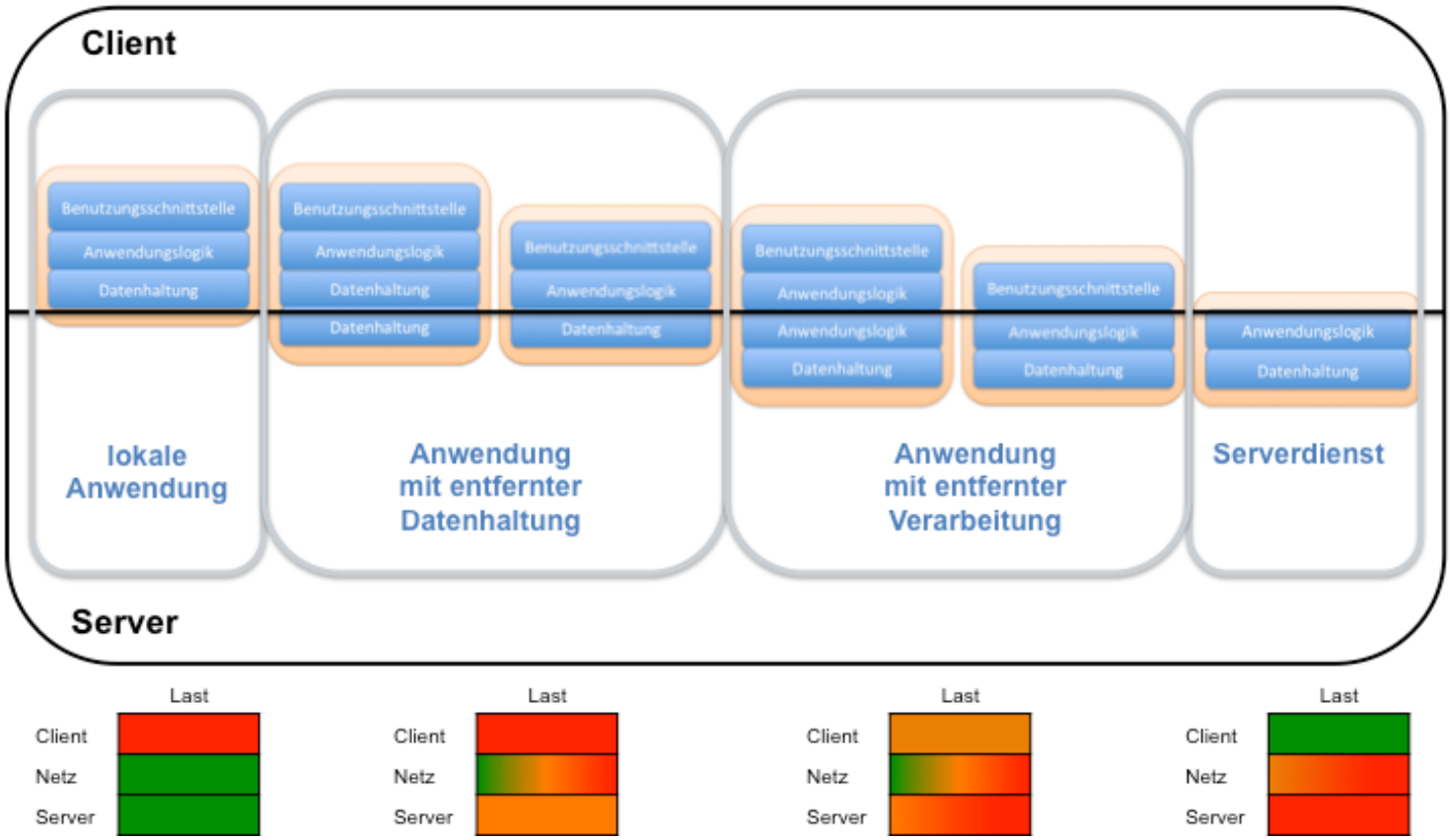
Indikatoren:

a) Qualitative Beurteilung der Vollständigkeit und Verständlichkeit
b) Bezieht sich die Produktinformation auf die aktuelle Produktversion?
c) Augenscheinprüfung der Korrektheit der Angaben (Angaben sind schlüssig / eingeschränkt schlüssig / nicht schlüssig)

Anhang A: Klassifikation von Anwendungssoftware

Klassifikation von Anwendungssoftware bezüglich der Softwarearchitektur

Fokus: Arbeitsteilung zwischen Client und Server



Erläuterungen:

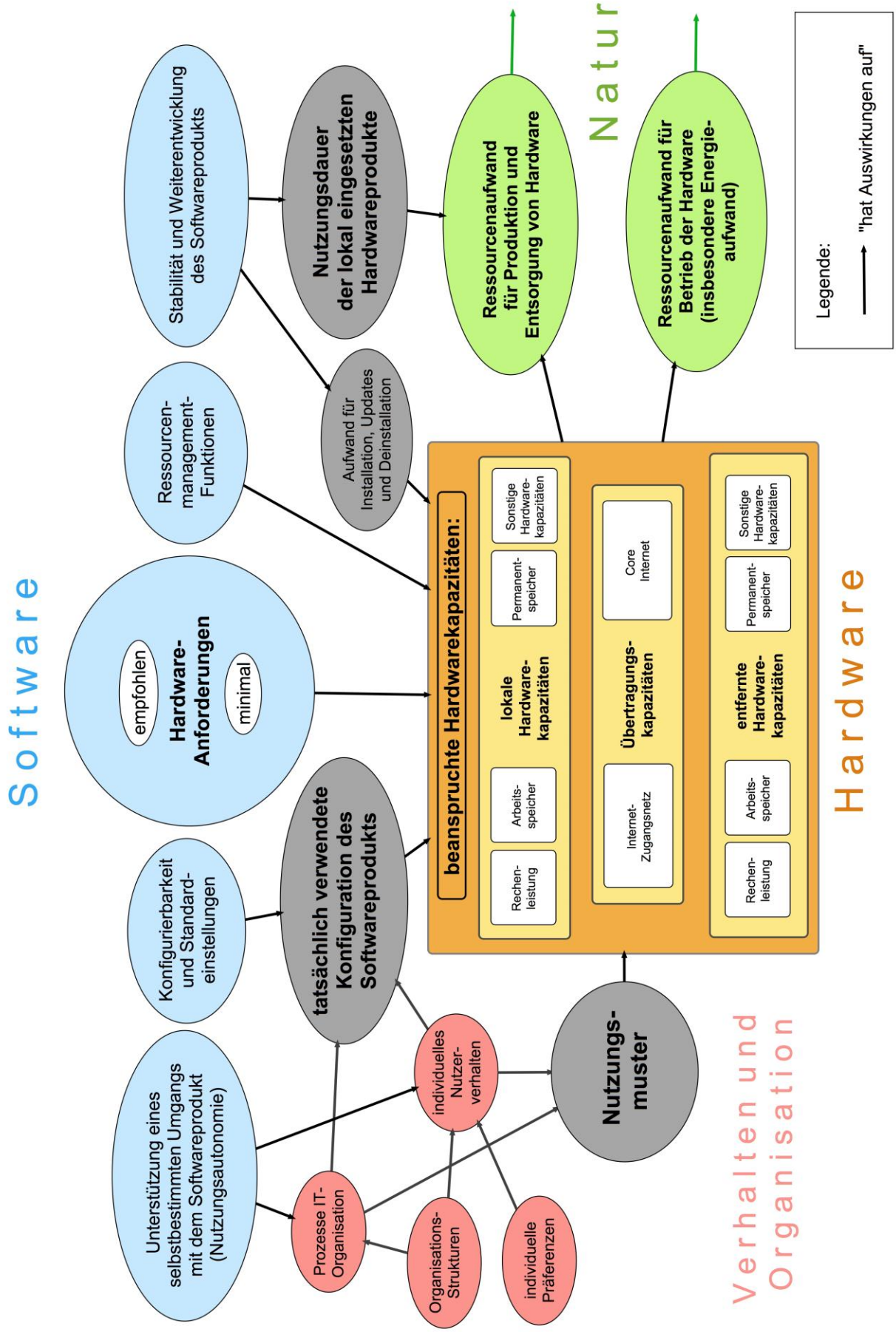
Benutzungsschnittstelle: Die Bestandteile des Softwareprodukts, die für die Interaktion mit Nutzenden zuständig sind.

Anwendungslogik: Der funktionale Kern des Softwareprodukts; beinhaltet alle Verarbeitungsmechanismen und Zugriffe auf die Datenhaltungsschicht.

Datenhaltung: Die Bestandteile des Softwareprodukts, die für das Speichern und Laden von Daten, z.B. in einer Datenbank, zuständig sind.

Die Farben stellen Anteile an der Gesamtlast dar, von 0 % (grün) bis 100 % (rot).

Anhang B: Wirkungsmodell



Anhang C: Glossar

Energieeffizienz: Allgemein die Menge an „nützlicher Arbeit“ dividiert durch den dabei anfallenden Energieaufwand. Im Kontext dieses Dokuments wird „nützliche Arbeit“ als erfolgreiche Ausführung von Standardnutzungsszenarien operationalisiert.

Hardware: Gesamtheit der für die Ausführung von Programmen, die Speicherung von Daten oder den Transport von Daten benötigten Sachgüter.

Hardwarekapazität: Quantifizierbare Eigenschaft eines Hardwaresystems, die die Grenze seiner Leistungsfähigkeit auf einer gegebenen Leistungsdimension darstellt (z.B. Arbeitsspeicherkapazität, Rechenkapazität, Bandbreite).

Hardwaresystem: Abgrenzbare Einheit von Hardware, die definierte Funktionen erbringt.

Indikator: Eine empirisch bestimmbare Größe, die Aufschluss über einen nicht direkt messbaren Sachverhalt gibt. Das Skalenniveau der in diesem Dokument vorgeschlagenen Indikatoren ist unterschiedlich. In einigen Fällen wird man mit einer Ordinalskala leben müssen (z.B. “ungenügend”, “genügend”, “gut”, “sehr gut” oder auch nur „erfüllt“, „nicht erfüllt“), um zu vermeiden, durch Verwendung einer Kardinalskala eine nicht vorhandene Präzision vorzuspiegeln.

Nutzungsmuster: Abstrahierte Form einer Sequenz von Interaktionen mit einem gegebenen Softwareprodukt.

Nutzungsszenario: Beschreibung eines Nutzungsmusters, in der Regel maschinell ausführbar.

Referenzsystem: Ein Hardwaresystem, das hinsichtlich seiner wichtigsten Kapazitäten (z.B. Arbeitsspeicher, Prozessorleistung) während einer festgelegten Zeitperiode (z.B. ein Jahr) als allgemein üblich definiert wird. Das Referenzsystem dient dazu, Indikatoren wie z.B. „minimaler lokaler Arbeitsspeicher“ relativ zu einer Referenzgröße (der aktuell „üblichen“ Arbeitsspeicherkapazität) ausdrücken zu können.

Ressource: Im Kontext dieses Dokuments eine natürliche Ressource, insbesondere ein Rohstoff, eine Energieform oder auch die Absorptionsfähigkeit eines Umweltmediums für Emissionen. Zur Abgrenzung gegen technische Ressourcen, insbesondere Hardwareressourcen, werden letztere hier präziser als „Hardwarekapazitäten“ bezeichnet. Da die Beanspruchung von Hardwarekapazitäten stets zur Beanspruchung natürlicher Ressourcen führt, ist diese Abgrenzung (die in letzter Konsequenz auf eine definitorisch schwierige Grenzziehung zwischen Ökosphäre und Technosphäre hinausläuft) hier nicht von entscheidender Bedeutung.

Ressourceneffizienz: Allgemein die Menge an „nützlicher Arbeit“ dividiert durch den dabei anfallenden Aufwand an Ressourcen. Im Kontext dieses Dokuments wird „nützliche Arbeit“ als erfolgreiche Ausführung von Standardnutzungsszenarien operationalisiert.

Software: Programme und Daten in digitaler Form.

Softwareprodukt: Eine abgrenzbare Einheit von Programmen und Daten, für die eine Nutzungslizenz angeboten wird.

Standardkonfiguration: Eine als Referenz definierte Menge von Bedingungen, unter denen ein gegebenes Softwareprodukt betrieben wird; sie umfasst die am Softwareprodukt während Installation oder Betrieb vorgenommenen Parametereinstellungen, die bereitgestellte Systemsoftware, ggf. weitere zum Betrieb benötigte Softwareprodukte sowie auf Hardwareebene das Referenzsystem.

Standardnutzungsszenario: Ein Nutzungsszenario, das zum Testen eines Softwareprodukts verwendet wird und möglichst repräsentativ für den üblichen Anwendungsfall sein soll.

Anhang D: Verwendete Literatur

Abdullah, Rusli; Abdullah, Salfarina; Din, Jamilah; Tee, Mcxin; others (2015): A Systematic Literature Review of Green Software Development in Collaborative Knowledge Management Environment. In: International Journal of Advanced Computer Technology (IJACT) 9, S. 136.

Abdullah, Rusli; Abdullah, Salfarina; Tee, Mcxin (2014): Web-based knowledge management model for managing and sharing green knowledge of software development in community of practice. In: Software Engineering Conference (MySEC), 2014 8th Malaysian. IEEE, S. 210–215.

Afgan, Naim Hamdia (2010): Sustainability paradigm: intelligent energy system. In: Sustainability 2 (12), S. 3812–3830.

Afzal, Shehla; Saleem, M. Faisal; Jan, Fahad; Ahmad, Mudassar (2013): A Review on Green Software Development in a Cloud Environment Regarding Software Development Life Cycle:(SDLC) Perspective. In: International Journal of Computer Trends and Technology (IJCTT) 4 (9), S. 3054–3058.

Agarwal, Shalabh; Nath, Asoke; Chowdhury, Dipayan (2012): Sustainable Approaches and Good Practices in Green Software Engineering. In: IJRRCS 3 (1), S. 1425–1428. Online verfügbar unter scholarlyexchange.org, zuletzt geprüft am 15.03.2012.

Ahmad, Ruzita; Baharom, Fauziah; Hussain, Azham (2014): A Systematic Literature Review on Sustainability Studies in Software Engineering. In: Proceedings of KMICe. Knowledge Management International Conference (KMICe) 2014. Malaysia, 12 – 15 August 2014.

Albertao, Felipe (2004): Sustainable Software Engineering. Carnegie Mellon University Silicon Valley. Online verfügbar unter www.scribd.com, zuletzt aktualisiert am 04.09.2008, zuletzt geprüft am 30.11.2010.

Albertao, Felipe; Xiao, Jing; Tian, Chunhua; Lu, Yu; Zhang, Kun Qiu; Liu, Cheng (2010): Measuring the Sustainability Performance of Software Projects. In: IEEE Computer Society (Hg.): 2010 IEEE 7th International Conference on e-Business Engineering (ICEBE 2010), Shanghai, China. Technical Committee on Electronic Commerce (TCEC), S. 369–373. Online verfügbar unter doi.ieeecomputersociety.org, zuletzt geprüft am 04.03.2011.

Amsel, Nadine; Ibrahim, Zaid; Malik, Amir; Tomlinson, Bill (2011): Toward sustainable software engineering (NIER track). In: Proceedings of the 33rd International Conference on Software Engineering. ACM, S. 976–979.

Ardito, Luca; Morisio, Maurizio (2014): Green IT - Available data and guidelines for reducing energy consumption in IT systems. In: Sustainable Computing: Informatics and Systems 4 (1), S. 24–32.

RAL-UZ 161, 2012-07: Basic Criteria for Award of the Environmental Label Energy-Conscious Data Centers. Online verfügbar unter www.eco-institut.de

Berkhout, Frans; Hertin, Julia (2001): Impacts of Information and Communication Technologies on Environmental Sustainability: speculations and evidence. Report to the OECD. Hg. v. Organisation for Economic Co-operation and Development OECD. Brighton. Online verfügbar unter www.oecd.org, zuletzt geprüft am 02.03.2011.

Bouwers, Eric; van Deursen, Arie; Visser, Joost (2013): Evaluating usefulness of software metrics: an industrial experience report. In: Proceedings of the 2013 International Conference on Software Engineering. IEEE Press, S. 921–930.

Bozzelli, Paolo; Gu, Qing; Lago, Patricia (2013): A systematic literature review on green software metrics. Technical Report: VU University Amsterdam.

Calero, C.; Bertoa, M.F; Angeles Moraga, M. (2013a): A systematic literature review for software sustainability measures. In: Green and Sustainable Software (GREENS), 2013 2nd International Workshop on, S. 46–53.

Calero, Coral; Bertoa, Manuel F.; Moraga, Maria Ángeles (2013b): Sustainability and Quality: Icing on the Cake. In: RE4SuSy@RE. Citeseer.

Calero, Coral; Moraga, M.; Bertoa, Manuel F. (2013c): Towards a software product sustainability model. In: arXiv preprint arXiv:1309.1640.

Calero, Coral; Moraga, Maria Ángeles; Bertoa, Manuel F.; Duboc, Leticia (2015): Green Software and Software Quality. In: Coral Calero und Mario Piattini (Hg.): Green in Software Engineering: Springer, S. 231–260.

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

Capra, Eugenio; Francalanci, Chiara; Slaughter, Sandra A. (2011): Is software green? Application development environments and energy efficiency in open source applications. In: Information and Software Technology 54, S. 60–71.

Dick, Markus; Naumann, Stefan (2010): Enhancing Software Engineering Processes towards Sustainable Software Product Design. In: Klaus Greve und Armin B. Cremers (Hg.): EnviroInfo 2010: Integration of Environmental Information in Europe. Proceedings of the 24th International Conference on Informatics for Environmental Protection, October 6 - 8, 2010, Cologne/Bonn, Germany. Aachen: Shaker, S. 706–715.

EPA ENERGY STAR (2014): ENERGY STAR Program Requirements Product Specification for Computers: Eligibility Criteria, Version 6.1. Environmental Protection Agency. Online verfügbar unter http://www.energystar.gov/sites/default/files/specs/Version%206%201%20Computers%20Final%20Program%20Requirements.pdf.

EPA Office of Air and Radiation, Climate Protection Partnerships Division (2015): National Awareness of ENERGY STAR for 2014. Analysis of CEE Household Survey. Hg. v. U.S. Environmental Protection Agency. Online verfügbar unter www.energystar.gov.

Erdmann, Lorenz; Hilty, Lorenz M.; Goodman, James; Arnfalk, Peter (2004): The Future Impact of ICTs on Environmental Sustainability. Technical Report EUR 21384 EN. Hg. v. Carlos Rodríguez Casal, Christine Van Wunnik, Luis Delgado Sancho, Jean Claude Burgelman und Paul Desruelle. European Commission; Joint Research Centre; IPTS - Institute for Prospective Technological Studies. Seville (Technical Report Series, EUR 21384 EN). Online verfügbar unter ftp.jrc.es, zuletzt geprüft am 26.07.2011.

Europäische Union (Hg.) (2011a): Beschluss der Kommission vom 6. Juni 2011 zur Festlegung der Umweltkriterien für die Vergabe des EU-Umweltzeichens für Notebooks. (Bekannt gegeben unter Aktenzeichen K(2011) 3736)Text von Bedeutung für den EWR. Online verfügbar unter eur-lex.europa.eu.

Europäische Union (2011b): Beschluss der Kommission vom 9. Juni 2011 zur Festlegung der Umweltkriterien für die Vergabe des EU-Umweltzeichens für Tischcomputer. (Bekannt gegeben unter Aktenzeichen K(2011) 3737)Text von Bedeutung für den EWR. Online verfügbar unter eur-lex.europa.eu.

Finkbeiner, Matthias; Schau, Erwin M.; Lehmann, Annekatrin; Traverso, Marzia (2010): Towards life cycle sustainability assessment. In: Sustainability 2 (10), S. 3309–3322. Online verfügbar unter www.mdpi.com.

Fujitsu Technology Solutions (Hg.) (2010): Green Label-Kategorien bei Fujitsu Technology Solutions. White Paper.

Fujitsu Technology Solutions (Hg.) (2012): Green Label Levels at Fujitsu Technology Solutions. White Paper. Online verfügbar unter globalsp.ts.fujitsu.com, zuletzt aktualisiert am 25.04.2012, zuletzt geprüft am 02.01.2013.

GeSI, Global e-Sustainability Initiative; The Climate Group (2008): SMART 2020: Enabling the low carbon economy in the information age.

Gröger, Jens; Köhn, Marina; Albers, Erik; Löhr, Patrik; Lohmann, Wolfgang; Naumann, Stefan (2015): Nachhaltige Software. Dokumentation des Fachgesprächs „Nachhaltige Software“ am 28.11.2014. Hg. v. Umweltbundesamt. Öko-Institut e.V. Dessau-Roßlau. Online verfügbar unter www.umweltbundesamt.de.

Gröger, Jens; Quack, Dietlinde; Grießhammer, Rainer; Gattermann, Marah (2013): TOP 100 - Umweltzeichen für klimarelevante Produkte: Freiburg. Online verfügbar unter www.ecodialog.de.

Held, Alexandra (2010): Entwicklung und Operationalisierung von Kriterien zur Bewertung der Nachhaltigkeit von Softwareprodukten. Abschlussarbeit zur Erlangung des akademischen Grades Master of Science eingereicht am Umwelt-Campus Birkenfeld. Masterarbeit. Fachhochschule Trier, Standort Umwelt-Campus Birkenfeld, Hoppstädten-Weiersbach. ISS Institut für Softwaresysteme in Wirtschaft, Umwelt und Verwaltung.

Hilty, Lorenz; Lohmann, Wolfgang; Behrendt, Siegfried; Evers-Wölk, Michaela; Fichter, Klaus; Hintemann, Ralph (2015): Grüne Software. Schlussbericht zum Vorhaben: Ermittlung und Erschließung von Umweltschutzpotenzialen der Informations- und Kommunikationstechnik (Green IT). Studie im Auftrag des Umweltbundesamtes, Berlin, Förderkennzeichen 3710 95 302/3 (im Druck).

Horne, Ralph E. (2009): Limits to labels: The role of eco-labels in the assessment of product sustainability and routes to sustainable consumption. In: International Journal of Consumer Studies 33 (2), S. 175–182. Online verfügbar unter 19-659-fall-2011.wiki.uml.edu.

Kern, Eva; Dick, Markus; Naumann, Stefan; Guldner, Achim; Johann, Timo (2013): Green Software and Green Software Engineering – Definitions, Measurements, and Quality Aspects. In: Lorenz M. Hilty, Bernard Aebischer, Göran Andersson und Wolfgang Lohmann (Hg.): ICT4S ICT for Sustainability. Proceedings of the First International Conference on Information and Communication Technologies for Sustainability, ETH Zurich, February 14-16, 2013. Zürich: ETH Zurich, University of Zurich and Empa, Swiss Federal Laboratories for Materials Science and Technology, S. 87–94. Online verfügbar unter e-collection.library.ethz.ch.

Koçak, Sedef Akınlı; Calienes, Giovanna Gonzales; Alptekin, Gülfem Işıklar; Bener, Ayşe Başar (2013): Requirements Prioritization Framework for Developing Green and Sustainable Software using ANP-based Decision Making. In: EnviroInfo, S. 327–335.

Koçak, Sedef Akınlı; Alptekin, Gülfem Işıklar; Bener, Ayşe Başar (2014): Evaluation of Software Product Quality Attributes and Environmental Attributes using ANP Decision Framework. In: Proceedings of the Third International Workshop on Requirement Engineering for Sustainable Systems (pp.

pp. 37-44). Karlskrona: Central Europe Workshop Proceedings. Online verfügbar unter ceur-ws.org

Lago, Patricia; Jansen, Toon; Jansen, Marten (2010): The service greenery-integrating sustainability in service oriented software. In: International Workshop on Software Research and Climate Change (WSRCC), co-located with ICSE, Bd. 2.

Lago, Patricia; Koçak, Sedef Akinli; Crnkovic, Ivica; Penzenstadler, Birgit (2015): Framing sustainability as a property of software quality. In: Communications of the ACM 58 (10), S. 70–78.

Lami, Giuseppe; Fabbrini, Fabrizio; Fusani Mario (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.

Mazijn, B.; Doom, R.; Peeters, H.; Vanhoutte, G.; Spillemaeckers, S.; Taverniers, L. et al. (2004): Ecological, Social and Economic Aspects of Integrated Product Policy. Integrated Product Assessment and the Development of the Label `Sustainable Development´ for Products. CP/20. SPSD II - Part I - Sustainable production and consumption patterns. Online verfügbar unter www.bernardmazijn.be.

Naumann, Stefan; Dick, Markus; Kern, Eva; Johann, Timo (2011): The GREENSOFT Model: A Reference Model for Green and Sustainable Software and its Engineering. In: SUSCOM 1 (4), S. 294–304. DOI: 10.1016/j.suscom.2011.06.004.

Penzenstadler, Birgit; Mahaux, Martin; Salinesi, Camille (2013): RE4SuSy: Requirements Engineering for Sustainable Systems. In: Journal of Systems and Software.

Prakash, Siddharth; Manhart, Andreas; Stratmann, Britta; Reintjes, Norbert (2008): Environmental product indicators and benchmarks in the context of environmental labels and declarations. Öko-Institut e.V.; Ökopol GmbH.

Schipper, Irene (2015): TCO Certified Smartphones versusFairphone. A comparison of sustainability criteria. Hg. v. GoodElectronics Network Südwind. Stichting Onderzoek Multinationale Ondernemingen (SOMO), Centre for Research on Multinational Cooperations, Netherlands. Amsterdam.

Schmidt, Benno (2014): Strategien für eine integrativ-nachhaltige Software-Entwicklung. Hochschule Bochum, Fachbereich Geodäsie. Bochum (14-02).

Schmidt, Benno; Wytzisk, Andreas; Plödereder, In E.; Grunske, L.; Schneider, E.; Ull, D.; others (2014): Software Engineering und Integrative Nachhaltigkeit. In: Erhard Plödereder, Lars Grunske, Eric Schneider und Dominic Ull (Hg.): INFORMATIK 2014. Big Data – Komplexität meistern. – Proceedings. 2. Workshop "Umweltinformatik zwischen Nachhaltigkeit und Wandel (UINW) / Environmental Informatics between Sustainability and Change"), 26.09.2014, im Rahmen der INFORMATIK 2014, 22.-26.09.2014 in Stuttgart. P-232: Lecture Notes in Informatics (LNI), S. 1935–1945.

Scholl, Gerd; Simshäuser, Ulla (2002): Machbarkeitsuntersuchung für Umweltzeichen-Analyse der Möglichkeiten zur Akzeptanzerhöhung des Umweltzeichens" Blauer Engel" für Haushaltsgroßgeräte (" Weiße Ware") bei potenziellen Zeichennehmern: Umweltbundesamt. Online verfügbar unter www.umweltbundesamt.de.

Stieß, Immanuel; Birzle-Harder, Barbara (2013): Der Blaue Engel-ein Klassiker mit Potenzial: eine empirische Studie zu Verbraucherakzeptanz und Marktdurchdringung des Umweltzeichens.

Sundblad, Yngve; Lind, Torbjörn; Rudling, Jan (2002): IT product requirements and certification from the users' perspective. In: Proceedings of WWDU 2002 Conference, S. 280–282. Online verfügbar unter http://cid.nada.kth.se/pdf/CID-176.pdf

International Standard ISO/IEC 25010:2011, 01.03.2011: Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models.

Taina, Juha (2011): Good, Bad, and Beautiful Software - In Search of Green Software Quality Factors. In: CEPIS UPGRADE XII (4), S. 22–27. Online verfügbar unter www.cepis.org, zuletzt geprüft am 09.01.2012.

TCO Development (Hg.) (2012): TCO Certified Notebooks 4.0. Online verfügbar unter tcodevelopment.com.

Teufel, J.; Rubik, F.; Scholl, G.; Stratmann, B.; Graulich, K.; Manhart, A. (2009): Untersuchung zur möglichen Ausgestaltung und Marktimplementierung eines Nachhaltigkeitslabels zur Verbraucherinformation. In: Project report of the Öko-Institut e. V. in cooperation with the Institut für ökologische Wirtschaftsforschung (IÖW) GmbH. Freiburg: Öko-Institut e. V. Online verfügbar unter download.ble.de.

Umweltbundesamt (Hg.) (2013): Ökodesign-Richtlinie <Computer und Computerserver>. Online verfügbar unter www.umweltbundesamt.de.

Vergabegrundlage für Umweltzeichen RAL-UZ 100, 2014-06: Vergabegrundlage für Car-Sharing.

Vergabegrundlage für Umweltzeichen RAL-UZ 78a, 2014-11: Vergabegrundlage für Umweltzeichen - Computer.

Walch, Isabelle (2015): Standard ECMA-370. TED - The ECO Declaration. Hg. v. ecma INTERNATIONAL.

Waller, Lars (2015): TCO Certified Desktops 5.0 - Critera Document and Certification Process. TCO Development. Online verfügbar unter tcodevelopment.com, zuletzt aktualisiert am 11.11.2015.

Warschun, Mirko; Rühle, Jens (2008): Zwischen ÖkoLabels, grüner Logistik und fairem Handel Lebensmitteleinzelhandel auf der Suche nach Wegen zur Nachhaltigkeit.