Ressourceneffiziente Programmierung

Wie Ressourcenschonung, Langlebigkeit und Nachhaltigkeit in der Softwareentwicklung berücksichtigt werden können

www.bitkom.org

bitkom

Herausgeber

Bitkom
Bundesverband Informationswirtschaft,
Telekommunikation und neue Medien e.V.
Albrechtstraße 10 | 10117 Berlin
T 030 27576-0
bitkom@bitkom.org
www.bitkom.org

Ansprechpartner

Dr. Frank Termer | Bitkom e.V.
T 030 27576-232 | f.termer@bitkom.org

Verantwortliches Bitkom-Gremium

AK Software Engineering & Software Architektur

Projektleitung

Dr. Leif Geiger | Yatta Solutions GmbH

Titelbild

© cyano66 – istockphoto.com

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Über die Autoren

Der vorliegende Leitfaden ist in enger Zusammenarbeit der folgenden Autoren entstanden. Wir danken allen Beteiligten herzlich für ihr Engagement.

Dr. Leif Geiger
ist Mitgründer und Produktmanager von ↗Yatta. Vor der Gründung von Yatta hat er an der TU Braunschweig Informatik studiert und an der Universität Kassel promoviert. Heute leitet er die Plattform-Entwicklung bei Yatta. Er ist Vorsitzender des Bitkom-Arbeitskreises für Software Engineering, im Vorstand des Kompetenzbereichs Software des Bitkom und engagiert sich in diversen Open-Source-Projekten, darunter für die Eclipse Plattform und dem Eclipse Marketplace Client. Auch heute versteht er sich noch als Wissenschaftler und Softwaretechniker.

Torsten Hopf
arbeitet als passionierter Softwareentwickler und -Architekt für ↗MHP Management- und IT-Beratung GmbH und ist ein Trusted Advisor für seine Kunden auf diesen Gebieten. Neben der Entwicklung innovativer Softwarelösungen hat er große Freude daran, sein Wissen mit Kollegen zu teilen und hat bereits zahlreiche interne Trainings sowie Vorlesungsreihen an der Dualen Hochschule Baden-Württemberg ausgearbeitet und durchgeführt. Seine Suche nach neuen interessanten Technologien ist angetrieben durch den Wunsch, skalierbare und zukunftssichere Software zu entwickeln.

Jacob Loring
ist für die ↗Code Intelligence GmbH im Business Development tätig. Bei dem Unternehmen handelt es sich um eine Ausgründung der Universität Bonn, das auf die Automatisierung von Software Security Tests spezialisiert ist, mit Hilfe von Feedback-based-Fuzzing. Im Rahmen seiner Tätigkeit vertritt er das Unternehmen in Gremien- und Verbandsangelegenheiten.

Mathias Renner
ist bei der ↗Frachtwerk GmbH als IT-Consultant, IT-Architekt und Product Owner tätig. Er unterstützt Kunden insb. im Kontext von cloud-native Daten-Plattformen mit Open Source. Dabei baut er als studierter Wirtschaftsinformatiker nicht nur die Brücke zwischen IT und Business, sondern auch zwischen IT und Ökologie, um die ökologischen Herausforderungen unserer Zeit im technologischen Kontext anzugehen. Sein Wissen teilt er zudem als Dozent an der Hochschule für Technik und Wirtschaft (HTW) Berlin im Master Wirtschaftsinformatik.

Johannes Rudolph
Johannes ist Mitgründer und CTO bei ↗**meshcloud**. Entstanden an der TU Darmstadt ist meshcloud eine multi-cloud governance Plattform, welche Organisationen erlaubt auf den endlosen Möglichkeiten der Cloud aufzubauen und dabei ihre digitale Souveränität beschützt. Gestartet als Software Engineer mit einer Leidenschaft für neue Technologien, begeistert er sich heute dafür wie diese mit organisatorischer Dynamik gewinnbringend »at scale« verknüpft werden kann.

Dr. Andreas Scharf
Dr. Andreas Scharf ist Technischer Leiter der Firma ↗**OctaVIA AG**. Er beschäftigt sich neben SAP spezifischen Themen vor allem mit Software Architekturen in verteilten Systemen sowie neuen Technologien im Allgemeinen. Neben der Leidenschaft für die Softwareentwicklung interessiert er sich auch für die Themen Organisationsentwicklung und Geschäftsmodellinnovation. In diesem Kontext bietet er als Dozent regelmäßig Vorlesungen an der Universität Kassel an. Er hat an der Universität Kassel Informatik studiert und im Fachgebiet Software Engineering promoviert.

Martin Schmidt
ist Data Scientist und Product Owner bei der ↗**DB Systel GmbH**, Dozent an der Digital Business University of Applied Sciences (DBU) sowie Assoziierter Forscher am Alexander von Humboldt Institut für Internet und Gesellschaft (HIIG). Der promovierte Umweltwissenschaftler beschäftigt sich zudem mit Nachhaltigkeit von und mit Digitalisierung.

Dr. Frank Termer
ist Bereichsleiter Software beim ↗**Bitkom e.V.** Nach seinem Studium der Wirtschaftsinformatik an der Otto-von-Guericke Universität Magdeburg war er ab 2006 als IT-Consultant tätig und führte Projekte im Geschäftsprozess- und IT-Servicemanagement durch. Hierbei betreute er vor allem Unternehmen aus den Bereichen Energiewirtschaft, Finanzdienstleistungen sowie der öffentlichen Verwaltung. Ab 2010 arbeitete er als wissenschaftlicher Mitarbeiter an der Technischen Universität Ilmenau im Fachgebiet Wirtschaftsinformatik für Dienstleistungen. Dort legte er seinen Forschungsschwerpunkt auf Fragen des strategischen IT-Managements. Während dieser Tätigkeit entstanden zahlreiche Publikationen zu den Themen IT-Agilität, Enterprise Architecture Management, Geschäftsprozessmanagement und Business-IT-Alignment. Er promovierte zum Thema »Determinanten der IT-Agilität«. Seit 2015 betreut er im Bitkom e.V. die Gremien des Kompetenzbereichs Software. Er konzipiert, organisiert und moderiert Gremienveranstaltungen, ist verantwortlich für die thematische Weiterentwicklung dieser Gremien sowie deren inhaltliche Positionierung innerhalb und außerhalb des Bitkom.

1 Warum sich mit dem Thema beschäftigen?

Der Klimawandel stellt eine der größten Herausforderungen der Menschheit für die nächsten Jahre und Jahrzehnte dar. Damit das Ziel der Klimaneutralität, das sich die EU als Langzeitklimaziel gemäß des ↗**Klimabkommens von Paris** gesetzt hat, erreicht werden kann, müssen sich alle Branchen und Industrien entsprechend aufstellen. Zunächst geraten hier Kohlekraftwerke, die Stahlproduktion oder auch Dieselfahrzeuge in den Blick, aber im Informations- und Telekommunikations (ITK)-Sektor liegt ebenfalls enormes Potential und somit besteht Handlungsbedarf.

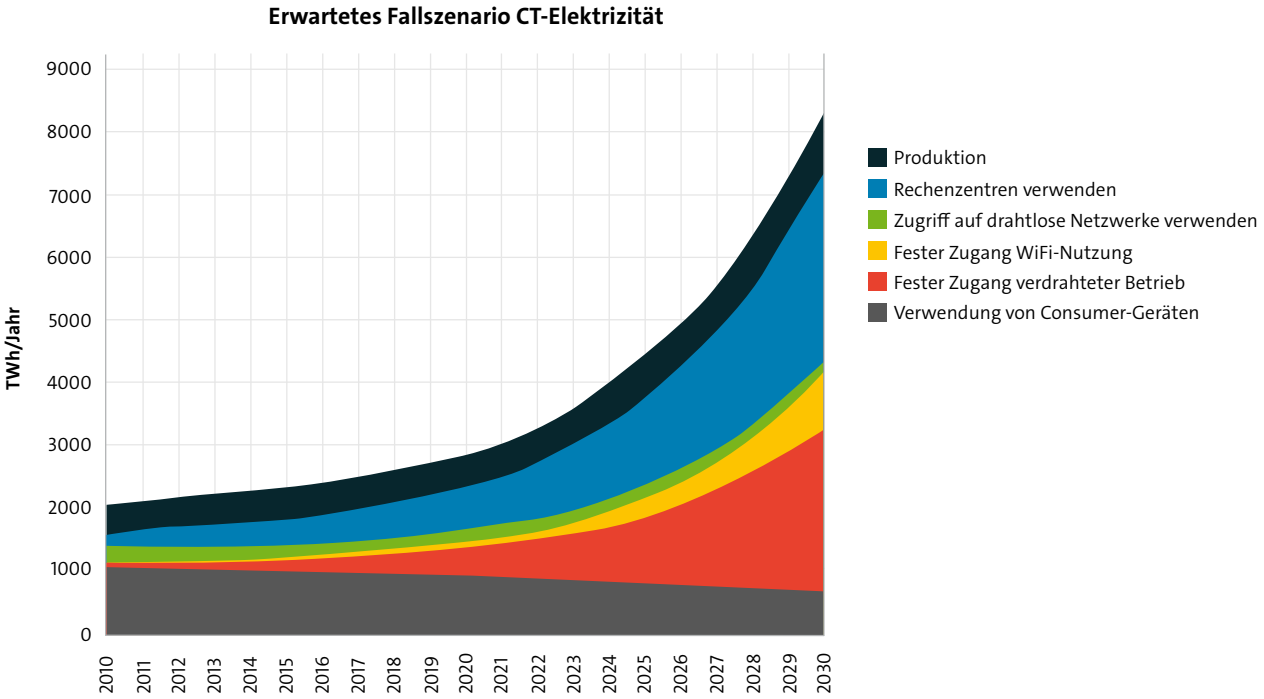


Abbildung 1: Szenario des Energieverbrauchs im Bereich Informationstechnologie, Quelle: Andrae und Edler (2015)

Ansteigender Trend: Software ist für einen zunehmenden CO2-Ausstoß verantwortlich

Verschiedene Studien und Untersuchungen kommen zu der Erkenntnis, dass der Anteil der ITK-Branche am weltweiten Energieverbrauch im Jahr 2020 zwischen 1% und 3,2% liegt (vgl. Jones 2018, vgl. Bitkom 2020). Prognosen gehen davon aus, dass dieser Anteil auf bis zu 23% im Jahr 2030 ansteigen kann (vgl. Andrae und Edler 2015, vgl. Jones 2018). Trotz dieser sehr hohen Zahlen wird aktuell die Nachhaltigkeit und Ressourcenschonung nur bei wenigen IT-Projekten beachtet. Eine Kurzumfrage unter Bitkom-Mitgliedern ergab (vgl. Abbildung 2), dass bei 23 Prozent der befragten Unternehmen das Thema Nachhaltigkeit in Softwareentwicklungsprojekten gar keine Rolle spielt. Bei 70 Prozent der Unternehmen spielt Nachhaltigkeit lediglich bei bis zu 25 Prozent der Softwareentwicklungsprojekte eine Rolle.

Bei wie viel Prozent der bisherigen Softwareentwicklungspojekte wurde Nachhaltigkeit bereits explizit beachtet?

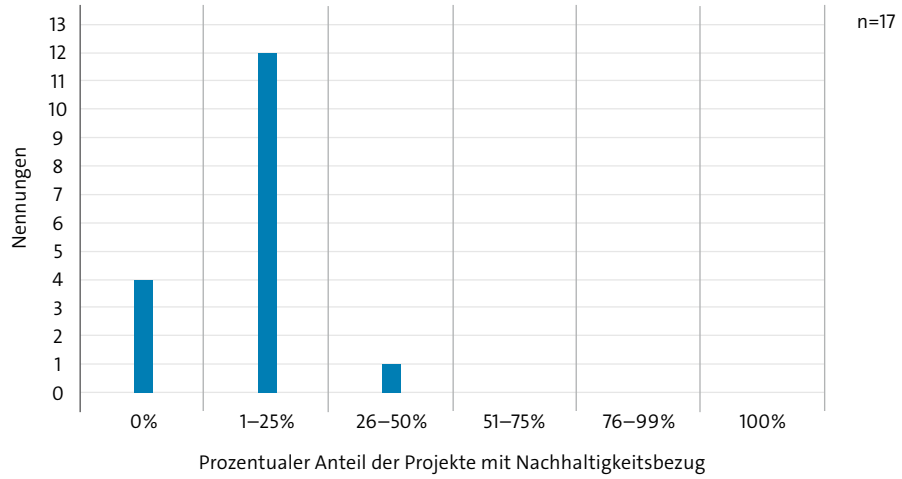


Abbildung 2: Ergebnisse einer Kurzumfrage zum Thema »Berücksichtigung von Nachhaltigkeit in Softwareentwicklungsprojekten«

Dieser Leitfaden soll einen Überblick über die Möglichkeiten der Berücksichtigung von Nachhaltigkeit, Langlebigkeit und Ressourceneffizienz bei Softwareentwicklungsprojekten bieten und gibt gleichzeitig eine aktive Hilfestellung, um verantwortlichen Personen den Einstieg in das Themenfeld der nachhaltigen Softwareentwicklung zu erleichtern. Für einen raschen Überblick liefert ein Cheat Sheet im Anhang konkrete Hinweise und Möglichkeiten, wie Ressourceneffizienz beim Software Engineering realisiert werden kann.

Begriffsdefinition

Unter **Nachhaltigkeit** verstehen wir hier die Definition aus dem Brundtland-Bericht (Hauff 1987):

»Dauerhafte Entwicklung ist Entwicklung, die die Bedürfnisse der Gegenwart befriedigt, ohne zu riskieren, dass künftige Generationen ihre eigenen Bedürfnisse nicht befriedigen können.«

Wir konzentrieren uns hier aber insbesondere auf Punkt 13 der UN-Nachhaltigkeitsziele »Massnahmen zum Klimaschutz« (vgl. Vereinte Nationen 2016).

Ein Leitfaden für ressourceneffiziente Programmierung

Gemäß des Greensoft-Referenzmodells (Naumann et al. 2011) wendet sich dieser Leitfaden insbesondere an Entwicklerinnen und Entwickler, Administratorinnen und Administratoren sowie Architektinnen und Architekten und deckt den kompletten Lebenszyklus eines Softwareprodukts von der Entwicklung bis zum End of Life ab. In ↗ **Kapitel 2** werden Ideen skizziert,

Denkanstöße gegeben, aber auch Handlungsempfehlungen aufgezeigt, wie das Potential für nachhaltige Softwareentwicklung in Organisationen identifiziert und umgesetzt werden kann.

Um nachhaltige Softwareentwicklung permanent im Unternehmen zu verankern, sind unterstützende organisatorische Maßnahmen ebenso erforderlich und zielführend. So wendet sich ↗**Kapitel 3** an die zugehörigen Entscheidungtragenden und zeigt auf, wie Anreizsysteme funktionieren können, um eine nachhaltige Organisation aufzubauen.

Der Energieverbrauch einer Software steht in Abhängigkeit zu weiteren nichtfunktionalen Anforderungen, wie bspw. Sicherheit, Qualität, Kosten und Geschwindigkeit. Die Auswahl der richtigen Maßnahmen und Werkzeuge zur Energiereduzierung von Software ist folglich immer projektabhängig. Es gilt, sich bei der Softwareentwicklung immer auf die Punkte zu konzentrieren, bei denen der größte Hebel zur Energieeinsparung liegt. Daher wird in ↗**Kapitel 4** beschrieben, wie gemessen werden kann, welchen Effekt einzelne Maßnahmen auf den Energieverbrauch haben.

In ↗**Kapitel 5** werden abschließend die gesetzlichen und gesellschaftlichen Rahmenbedingungen betrachtet und es wird ein Ausblick auf zukünftige Entwicklungen gegeben.

2 Potenzialanalyse

Jedes Softwareentwicklungsprojekt ist anders: mal wird auf der grünen Wiese angefangen, mal sind Änderungen an einem komplexen Bestandssystem notwendig. Welche Richtlinien können generell beachtet werden, um ressourceneffizient zu programmieren?

2.1 Entwicklung

Measure what matters

1) Finde die Einflussgrößen
Generell gilt es sich auf die Aspekte zu konzentrieren, die den größten Hebel haben. Was sind die entscheidenden Technologien, die im Projekt je nach Ressourcenbedarf ausgewählt werden können? Wo wird die meiste CPU-Zeit benötigt, der meiste Arbeitsspeicher gebraucht, der meiste Netzwerk-Traffic erzeugt? Diese Fragen sind üblicherweise gute Startpunkte, um ressourceneffizient zu entwickeln.

2) Miss den Energieverbrauch
Ob bei der Softwareentwicklung wirklich Energie eingespart wurde, kann nur festgestellt werden, wenn der Energiebedarf einer Anwendung auch gemessen wird. Es kann sinnvoll sein, früh im Projekt Metriken einzuführen oder Frameworks einzusetzen um den Energiebedarf von kritischen Applikationsteilen zu untersuchen. Auf der Webseite ↗»Energy Effiency Across Programming Languages« wird bspw. ein komplettes Framework zur Beurteilung von Programmiersprachen bereit. Diese Webseite basiert auf der Arbeit von Pereira et al. (2017). Kapitel 4 beschäftigt sich im Detail mit Messmethoden, insbesondere für die Cloud.

Technologie

1) Wähle die Programmiersprache mit Bedacht
Einen größeren Einfluss auf die Energieeffizienz hat die Auswahl der Programmiersprache. Studien zeigen, dass in den meisten Fällen die schnellste Sprache für einen Anwendungsfall oft auch die energieeffizienteste Sprache ist. Welches die geeignete Sprache ist, hängt aber stark vom Anwendungsfall ab. Daher gilt, wenn ein neues Projekt gestartet wird und noch die Möglichkeit besteht, die Programmiersprache zu wählen, so kann es durchaus sinnvoll sein, auch die Energieeffizienz der Sprache zu berücksichtigen. Ein erster Überblick über die Energieeffizienz von unterschiedlichen Sprachen kann bei Cassel (2018) und Pereira et al. (2017) nachgelesen werden.

2) Wäge den Einsatz von Bibliotheken und Frameworks gut ab
Bibliotheken und Frameworks bieten zumeist optimierte Lösungen für verbreitete Probleme. Daher kann der Einsatz insbesondere von energieoptimierten Bibliotheken zu großen Einsparungen beim Energieverbrauch führen. So sind Bibliotheken, die einen Algorithmus möglichst effizient umsetzen, häufig auch energiesparender als eine ineffiziente Eigenentwicklung, da so

insgesamt weniger Rechenzeit verbraucht wird. Generell ist allerdings auch zu beachten, dass zur Lösung eines kleinen randständigen Problems die Integration eines komplexen Frameworks aus Nachhaltigkeitsaspekten oft nicht sinnvoll ist.

3) Lege passende Datenformate fest
Der Auswahl effizienter Datenformate kommt auf Grund weiter steigender Vernetzung und erfasster Datenmengen eine große Bedeutung zu. Effiziente Datenformate müssen immer anhand der zugrundeliegenden Quelldaten, deren Struktur und den Anforderungen der Verarbeitung (z.B. Versionierung, Metadaten oder Integrität) gewählt werden. Besonders deutlich wird der Einfluss auf die Ressourceneffizienz an einem Beispiel aus dem Bereich Internet of Things (IoT) für die Übertragung von Sensordaten. Hier benötigen binäre im Vergleich zu text-basierten Datenformaten deutlich weniger Speicherplatz, weniger Bandbreite für die Übermittlung und weniger Rechenzeit zur Verarbeitung. Dies wiederum hat direkten Einfluss auf den gesamten Energiebedarf eines Systems.

Design for Sustainability

1) Entwirf eine energieeffiziente Architektur
Software sollte bereits vom Entwurf an so gestaltet sein, dass sie energieeffizient genutzt werden kann. Kapitel 2.2. beleuchtet diesen Aspekt im Detail.

Laufzeit

1) Hinterfrage die Notwendigkeit von Berechnungen
Der Großteil der Energie lässt sich bei den meisten Projekten zur Laufzeit einsparen. Hier gilt: Code der nicht ausgeführt wird, verbraucht auch keine Energie. Daher sollte eine Applikation idealerweise so gestaltet sein, dass Berechnungen nicht »auf Vorrat« durchgeführt werden, und keine unnötigen Prozesse vorgehalten werden.

2) Ermögliche Anwenderinnen und Anwendern eine energiearme Nutzung
Sinnvoll ist es auch, dass Anwenderinnen und Anwendern von Software die Möglichkeit gegeben wird, bei der Benutzung der Applikation auf den Energieverbrauch zu achten. So kann beispielsweise die Möglichkeit gegeben werden, nicht benötigte Features abzuschalten oder gar rechenintensive Operationen erst dann durchzuführen, wenn Ökostrom zur Verfügung steht.

3) Minimiere den Datenfluss
Faktoren, wie CPU-Zeit und Speicherverbrauch, beeinflussen den Energieverbrauch zur Laufzeit. Insbesondere ist aber auch das Verschieben von Daten (z.B. vom Speicher zur Festplatte oder über das Netzwerk) ein Aspekt, der viel Energie verbraucht und oft gut zu optimieren ist. Es ist daher oft eine gute Idee mit Caches oder Buffers zu arbeiten und den Datenfluss generell gering zu halten.

4) Betreibe Anwendungen klimafreundlich
Wenn möglich, sollte darauf geachtet werden, dass der benötigte Strom beim Betrieb von Software klimafreundlich produziert wird. Verschiedene Cloud-Anbieter setzen beispielsweise auf erneuerbare Energien zum Betrieb ihrer Rechenzentren (siehe Kapitel 4).

CO2-Rechner von Cloud Anbietern

Manche Cloud-Provider bieten für Kunden einen CO2-Rechner an, mit dem der kundenspezifische Ressourcenverbrauch geschätzt wird. Hierfür werden die CO2-Emissionen berechnet, die durch den Rechenzentrumsbetrieb für einen Kunden anteilig entstehen. Mittels monatlicher bzw. jährlicher Berichte kann damit der eigene CO2-Fußabdruck beobachtet werden. Diese Berichte können auch Grundlage für die unternehmenseigene Berichterstattung sein. Manche CO2-Rechner unterstützen ebenfalls bei der Definition von Einsparzielen und ermöglichen die Verfolgung der Zielerreichung.

Entwicklungsumgebung

1) Konfiguriere die CI/CD-Pipeline für eine ressourcenschonende Nutzung
Auch bei der Softwareentwicklung selbst, sollte der Ressourcenverbrauch im Blick behalten werden. Mit inkrementellen Compilern, automatischer Testausführung auf dem Continuous-Integration-System, Continuous Deployment etc. löst ein Tastenanschlag bzw. ein Commit eines Entwicklers oder einer Entwicklerin gegebenenfalls eine Vielzahl an Hintergrundprozessen aus, die wiederum Energie verbrauchen. Diese Errungenschaften der modernen Softwareentwicklung sollten keinesfalls aufgegeben werden, da sie einen enormen Produktivitätsgewinn bringen. Dennoch sollte auf eine geeignete Konfiguration der CI/CD-Pipeline geachtet werden. Wie oft sollten die rechenaufwendigen Integrationstests auf dem CI-Server durchgeführt werden? Ist ein automatischer Durchlauf der kompletten Testsuite bei jedem behobenen Typo notwendig?

2) Wähle energieeffiziente Entwicklungswerkzeuge
Zu guter Letzt kann auch durch die geschickte Wahl der Entwicklungswerkzeuge Energie eingespart werden. Fundierte Untersuchungen hierzu sind noch selten bzw. veralten diese sehr schnell. So haben Amsel et al. (2011) beispielsweise festgestellt, dass der Internet Explorer effizienter als Chrome und Firefox ist. Hier wäre eine kontinuierliche Untersuchung z.B. im Rahmen des blauen Engels (siehe ↗**Kapitel 5**) hilfreich.

Einige Anbieter von Softwareentwicklungswerkzeugen und entsprechenden Services beachten bereits Aspekte des Klimaschutzes. So optimiert zum Beispiel der Editor Notepad++ (↗**https://notepad-plus-plus.org/**) aktiv Routinen, wodurch die CPU-Auslastung reduziert wird. Die Suchmaschine Ecosia (↗**https://info.ecosia.org/what**) verwendet die Werbeeinnahmen bei Suchabfragen zum Pflanzen von Bäumen.

2.2 Architektur

Eine Aufgabe im Bereich Software Architektur besteht darin zu kontrollieren, ob nichtfunktionale Anforderungen und damit verbundene Qualitätsziele eingehalten werden. Der Aspekt der Nachhaltigkeit bzw. Umweltfreundlichkeit wird im Bereich der Softwarequalität bisher noch unzureichend berücksichtigt. Bei den von der ↗**ISO/IEC-Norm 25010** aufgestellten Kriterien für Softwarequalität fehlt Nachhaltigkeit komplett (vgl. Raturi et al 2014). Lediglich der Ressourcenverbrauch wird hier aufgezählt, allerdings auch aus Sicht der Leistungseffizienz des Systems.

1) Hinterfrage die Dimensionierung deiner Infrastruktur

Das ↗**Wirthsche Gesetz** besagt, dass immer komplexere Software, die Effizienzsteigerungen in der Hardware überkompensiert. Eine Ursache hierfür ist, dass Softwaresysteme häufig anhand von Erwartungen dimensioniert werden, anstatt sich auf reale Daten zu stützen (Lago 2018). Dabei lässt sich eine Tendenz zur Überdimensionierung der Umgebung auf Grund von zu hohen Erwartungen in den Nutzerzahlen beobachten. Nicht selten wird ein Mengengerüst über mehrere Jahre hochgerechnet und anschließend eine Infrastruktur bereitgestellt, die von Anfang an die erwartete maximale Nutzerzahl unterstützt (vgl. St. Laurent 2018). Es liegt auf der Hand, dass durch ein solches Vorgehen viele Ressourcen durch always-on-Systeme verschwendet werden.

2) Berücksichtige den Energieverbrauch von Beginn an

Eine weitere Herausforderung liegt darin, dass es schwierig ist, die direkten Auswirkungen einer Software auf den Energieverbrauch eines Gesamtsystems festzustellen (vgl. Ardito et al. 2015). Messungen zum Energieverbrauch eines Gerätes mit unterschiedlicher Software, oder die Betrachtung einzelner Algorithmen oder Funktionen auf Code-Ebene scheinen hingegen wenig praktikabel. Dies macht es notwendig, dass die Nachhaltigkeit einer Architektur von Beginn an mit berücksichtigt wird.

3) Behandle Nachhaltigkeit als nicht-funktionale Anforderungen

Um Nachhaltigkeit in eine Architektur zu bringen, sollten nichtfunktionale Anforderungen bereits in der Anforderungsanalyse betrachtet und erfasst werden. Ein mögliches Vorgehen hierzu bildet das Sustainability non-functional-requirements framework (vgl. Raturi et al. 2014).

Nummer	Anforderung	Begründung
#1	Die Ergebnisse von komplexen Datenbankabfragen sollen gecached und nicht bei jeder Anfrage neu berechnet werden.	Durch das Einfügen einer Cache-Schicht, die für regelmäßige Anfragen genutzt wird, kann die CPU-Last auf den Datenbankservern und damit deren Energieverbrauch stark reduziert werden.

Nummer	Anforderung	Begründung
#2	Batch-Prozesse sollen nur während ihrer Ausführung Energie verbrauchen.	Durch das Ausführen von rechenintensiven Batch-Prozessen in getrennten Containern/ Berechnungseinheiten kann der Energieverbrauch der Always-on-Systeme reduziert werden.
...	...	...

Tabelle 1: Beispiele, wie Nachhaltigkeit in nicht-funktionalen Anforderungen berücksichtig werden kann

Diese nicht-funktionalen Nachhaltigkeitsanforderungen sollten im weiteren Entwurf der Architektur nach den gleichen Kriterien behandelt und gewichtet werden, wie ihre nicht auf Nachhaltigkeit bezogenen Äquivalente.

4) Lege Architektur-Constraints fest

Durch entsprechende Architektur-Prinzipien (vgl. The Open Group 2018) können in der Architektur Constraints festgelegt werden, die bei allen weiteren Architekturentscheidungen mit einbezogen werden müssen. Dabei kann frühzeitig in die Architektur verankert werden, dass eine Skalierung der Software- und Hardware-Komponenten nach Bedarf möglich ist. Der Energieverbrauch einer entwickelten Softwarelösung lässt sich drastisch reduzieren, wenn diese dynamisch zur Laufzeit durch das Hinzufügen oder Entfernen weiterer Instanzen an den tatsächlichen Bedarf angepasst wird. Entgegen dem sonst verbreiteten Mantra »größer zu denken« gilt es hier zukünftig »kleiner zu denken« und entsprechende Entscheidungen zu treffen. Ein wichtiges Ziel muss sein, Dynamik in die Dimensionierung von Ressourcen zu bringen, um einem Überdimensionieren aus Freigabegründen (»Jetzt-oder-nie«-Effekt) entgegenzuwirken.

5) Entkopple Softwarekomponenten voneinander

Durch eine strikte Entkopplung der Softwarekomponenten kann nicht nur die Verfügbarkeit und Resilienz verbessert werden, sondern auch der Ressourcenverbrauch positiv beeinflusst werden. Mehrere kleinere Services, welche mit jeweils niedrigen Hardwareanforderungen auskommen, können sparsamer und dennoch insgesamt leistungsfähiger sein, als ein klassischer Monolith, der auf einem oder gar mehreren Servern betrieben wird.

Durch die Entkopplung einzelner Komponenten und Verarbeitungsschritte kann auch der Energieverbrauch einer Softwarelösung drastisch verringert werden. Ein wiederkehrender, rechenintensiver Task (beispielsweise die abendliche Analyse aller Verkaufsergebnisse einer Supermarktkette) erhöht zum Beispiel die Hardware-Anforderungen an das Gesamtsystem in welchem er ausgeführt wird. Wenn das System ansonsten allerdings wenig rechenintensiv ist, geht viel Energie verloren, weil die Hardware auf diesen Edge-Case ausgerichtet ist. Durch ein Herauslösen eines solchen Task in eine eigene (bspw. containerized) Umgebung, die nur während der Ausführung des Tasks verfügbar gemacht wird, kann eine große Menge an Energie eingespart werden. Das eigentliche System und damit die durchgehend zu betreibende Hardware wird kleiner und effizienter.

6) Ziehe ein reaktives Modell in Betracht

Wird eine Webapplikation mit HTTP-Schnittstelle angeboten, sollte überlegt werden, ob diese »reaktiv« gestaltet sein sollte, anstatt das bewährte »One-Thread-per-Request-Modell« zu verwenden. Die Verwendung einer HTTP-Schnittstelle führt zwar zu einer höheren Komplexität und hebt die Einstiegshürde an, allerdings bietet es zahlreiche Vorteile. Neben einer erhöhten Antwortbereitschaft, Widerstandsfähigkeit und Elastizität (vgl. Bonér et al 2014), wird dem reaktiven Modell auch eine bessere Auslastung der Hardware attestiert (vgl. Singh 2020). Statt dutzender bzw. hunderter Threads mit langen i/o-waits, kann die Kommunikation Event-basiert über wenige (häufig einen) Thread erfolgen.

Zusammenfassung

Folge Punkte lassen sich nunmehr zusammenfassen:

- Nachhaltigkeit sollte von Anfang als nichtfunktionale Anforderung in ein Projekt integriert werden.
- Energie kann durch das Sicherstellen einer intelligenten Skalierung und mittels dynamischer Infrastruktur-Provisionierung eingespart werden.
- Eine möglichst lose Kopplung ist fundamental, um Potentiale zur Energieeinsparung aufdecken und nutzen zu können.
- Reaktive Architekturen und Programmiermodelle können dazu beitragen, vorhandene Ressourcen effizienter zu nutzen, um horizontale Skalierung und die damit verbundenen Kosten zu vermeiden.

3 Implementierung

Technische Potentiale und Umsetzungsmöglichkeiten für ressourceneffiziente Programmierung, die im ↗Kapitel 2 beleuchtet wurden, sind nur ein Aspekt einer ganzheitlichen Nachhaltigkeitsbetrachtung. Eine mindestens genauso wichtige Schlüsselkomponente zur erfolgreichen Umsetzung ressourceneffizienter Programmierung ist der organisatorische Rahmen, in dem Softwareentwicklung und Softwarebetrieb stattfinden. In diesem Kapitel sollen insbesondere Projektverantwortliche und Führungskräfte einen Überblick erhalten, wie Ressourceneffizienz in das Zielsystem eines Projekts integriert werden kann.

Erkenne Zielkonflikte frühzeitig
Projekte zur Entwicklung und Betrieb von Software sind sehr stark durch Zielkonflikte geprägt. Praktikerinnen und Praktiker aus dem Entwicklungsalltag kennen die konkreten Manifestationen des »magischen Zieldreiecks« von Zeit, Kosten und Qualität nur zu gut in Form von Fragen wie: »Sollen wir diesen Bug fixen oder lieber ein neues Feature entwickeln?« oder auch mit direktem Bezug zur Ressourceneffizienz: »Optimieren wir die Software oder nehmen wir einfach einen größeren Server?« Diese Zielkonflikte sind frühzeitig in Projekten aufzudecken und bewusst zu diskutieren.

Nutze Wechselwirkungen bewusst
Um Ressourceneffizienz und Nachhaltigkeit in Projekte integrieren zu können, dürfen diese Aspekte nicht als eigenständige, weitere Ziele aufgefasst werden. Die Auswirkungen ressourceneffizienter Programmierung auf etablierte Projektziele wie z.B. Kundenzufriedenheit und Betriebskosten sind stattdessen transparent aufzuzeigen und explizit zu berücksichtigen. Dieser Ansatz lässt sich deshalb auch einfach mit formalisierten Priorisierung-Frameworks wie z.B. ↗RICE integrieren.

3.1 Transparenz für alle Projektbeteiligten herstellen

Essentielle Grundlage für die Berücksichtigung von Ressourceneffizienz in allen Phasen eines Softwareprojekts ist die Verfügbarkeit einer gemeinsamen Datengrundlage zum tatsächlichen Ressourcenverbrauch in der damit erreichten Leistung des Projekts. In ↗Kapitel 4 wird aufgezeigt wie die dafür erforderliche Messbarkeit methodisch umgesetzt werden kann. Auf dieser Basis aufbauend müssen Projektverantwortliche sicherstellen, dass die erfassten Daten auch allen Projektbeteiligten in verständlicher Form zugänglich gemacht werden können. Dies kann z.B. in Form von Metriken und darauf aufbauender Dashboards erfolgen.

3.2 You build it, you run it, and run it well!

DevOps ist eine erfolgversprechende Grundlage für die Berücksichtigung von Ressourceneffizienz im Entwicklungs- und Betriebsprozess von Software. Die enge Verzahnung von Entwicklung und Betrieb gibt Entwicklerinnen und Entwicklern direkten Einblick in die Betriebsmetriken und

damit auch über die tatsächliche Ressourceneffizienz im produktiven Einsatz. DevOps-Teams tragen die »End-to-End«-Verantwortung für Entwicklung und Betrieb des Systems.

Die Kombination verschiedener Skillsets eröffnet neue Potentiale zur Verbesserung der Ressourceneffizienz (z.B. durch bessere Software-Hardware-Abstimmung) und kürzere Feedback-Zyklen bei der Umsetzung und Bewertung des Erfolgs von Maßnahmen zur Verbesserung der Ressourceneffizienz. Wird DevOps noch um den Aspekt des Business ergänzt (Stichwort: BizDevOps) kann auch die Ressourceneffizienz ganz konkret als Business Enabler thematisiert werden, z.B. wenn durch die Unterstützung älterer Hardware neue Kundensegmente erschlossen werden können oder eine verbesserte Skalierung der Betriebskosten neue Geschäftsmodelle ökonomisch rentabel machen.

3.3 Entscheidungsträger enablen – Verantwortung gemeinsam tragen

In der Softwareprojekt-Praxis wird der voran beschriebene Zielkonflikt zwischen dem Aufwenden von Entwicklungs-Zeit für neue Features oder dem Optimieren technischer Ressourceneffizienz häufig zugunsten Ersterem entschieden. Zur verbesserten Verständigung zwischen Entscheidungtragenden (häufig Stakeholder von Business-Seite) und technischen Teams werden in der Praxis bereits Konzepte wie »technical debt« erfolgreich angewandt. Die Kernidee dahinter ist, den Entscheidungtragenden Informationen über die Auswirkungen zukünftiger (und vergangener) Entscheidungen aufzuzeigen. Analog zu dieser Idee schafft Transparenz über die Ressourceneffizienz (siehe ↗**Kapitel 3.1**) eine gemeinsame Verständigungsgrundlage für verschiedene Stakeholder.

Die Verantwortung für die Umsetzung von Maßnahmen zur Ressourceneffizienz liegt damit nicht allein bei den Software-Entwicklerinnen und -Entwicklern, sondern auch bei den Entscheidungtragenden, welche die entsprechenden Maßnahmen priorisieren müssen. Transparenz über die Ressourceneffizienz von Maßnahmen ermöglicht es damit diese Entscheidungen als gemeinsame Verantwortung zu tragen.

3.4 Incentivierung überprüfen

Die Umsetzung von Ressourceneffizienz kann durch das Setzen positiver Anreize, bzw. das Vermeiden negativ wirkender Anreize, begünstigt werden. Organisatorische Anreize sollten dahingehend überprüft werden. Die vorigen Abschnitte betrachteten bereits Aspekte der Transparenz, End-to-End-Verantwortung und Priorisierung. Zur Flankierung dieser Maßnahmen können z.B. explizite positive Anreize wie Boni oder Incentive-Events wie Hackathons genutzt werden. Für die Vermeidung negativer Anreize sollte insbesondere darauf geachtet werden, dass Projektteams auch End-to-End-Verantwortung für die Ressourceneffizienz tragen können. Eine häufige Herausforderung ist hier die getrennte Budgetierung von Entwicklungs- und Betriebskosten, die diesem Ziel entgegensteht.

4 Messbarkeit

Nachhaltigkeit in der Softwareentwicklung hat viele technische Aspekte. Eine der wichtigsten Metriken hierfür ist die Energieeffizienz. Zum Beispiel der Stromverbrauch oder die Rechenleistung, die eine Anwendung benötigt. Doch auch Design- und Architektur-Aspekte spielen eine Rolle. Wie sind zum Beispiel einzelne Module integriert? Lassen sie sich aus der Ferne warten, ohne das die Hardware ausgetauscht werden muss? Wie lange soll die Software insgesamt im Einsatz sein? Das alles wirkt sich auf die CO2-Bilanz einer Software aus.

1) Mach Messen so einfach wie möglich

Es gibt daher viele verschiedene Metriken, wie man Nachhaltigkeit in der Softwareentwicklung berechnen kann. Welche Metriken ausgewählt werden und wie diese ermittelt werden können, darüber ließe sich lange streiten. Die Gefahr ist deshalb groß, sich in einer unproduktiven Diskussion zu verstricken. Wie in der IT-Sicherheit gilt daher das Prinzip: »Komplexität ist der Feind der nachhaltigen Softwareentwicklung«. Metriken und zugehörige Messvorschriften sollten daher so einfach wie möglich gestaltet werden, um die Anwendbarkeit in der Praxis zu erleichtern.

2) Arbeite mit Proxy-Werten

Ein gängiger Weg der Messung in der Praxis ist die Nutzung von Proxy-Werten, das heißt von Werten, die ein zu messenden Ziel nicht direkt, sondern indirekt über einen leichter zu ermittelnden Wert abbilden können. Ein Proxy-Wert für den CO2-Ausstoß, der sich besonders leicht berechnen lässt, ist zum Beispiel der Stromverbrauch. Dieser kann zum einen direkt ermittelt werden, oder aber auch indirekt bspw. durch die Angaben auf der Rechnung des Stromanbieters. Gleichzeitig ist für alle Beteiligten plausibel, das ein höherer Stromverbrauch in den meisten Fällen auch mit einem höheren CO2-Ausstoß verbunden ist.

Weitere Proxy-Werte, die in sich in der Praxis bewährt haben, sind zum Beispiel die CPU-Verarbeitung, der Speicherzugriff, der Input oder der Output eines Systems, oder die Menge des Datenverkehrs. Alternativ kann eine Orientierung auch an der Höhe der Rechnung eines Cloud-Anbieters erfolgen. Grundsätzlich gilt auch hier die Faustregel: »Je geringer die Kosten sind, desto weniger Strom wurde verbraucht«. Denn wenn der Stromverbrauch, oder die CPU-Verarbeitung niedrig ausfällt, dann sinken in der Tendenz auch die Betriebskosten einer Software. Maßnahmen, welche die Softwareentwicklung nachhaltiger gestalten wollen, wirken sich daher in vielen Fällen auch auf den nachhaltigen Betrieb der Software aus.

3) Denke die ganze Lieferkette mit

Im Zuge einer ganzheitlichen Betrachtung eines nachhaltigen Betriebs von IT-Systemen gilt es nicht nur den Stromverbrauch der eigenen Server zu messen, sondern auch die Rechenzeit auf Kunden- bzw. Nutzerseite im Blick zu haben und mit zu berücksichtigen. Bei Webservices kann diese Rechenleistung und der Stromverbrauch zum Teil massiv zum Gesamt-Energieverbrauch beitragen. Außerdem gibt es Fälle, die diese Rechnung verzerren. Zum Beispiel muss bedacht werden, aus welchen Energiequellen der Strom für den Betrieb der Software gewonnen wird. Hier kann durch die Wahl eines Energieanbieters, der Strom aus erneuerbaren Energiequellen bereitstellt, ein positiver Effekt auf den Gesamt-CO2-Fußabdruck eines IT-Systems erreicht werden.

5 Gesellschaftlicher und Politischer Rahmen

Als Unternehmen, Software-Architekt:in oder Software-Entwickler:in ist man bei der Umsetzung von ressourceneffizienter Programmierung nicht allein. Klare Richtlinien und Handlungsvorschriften durch die Gesellschaft und die Politik sind hilfreich, um das ganzheitliche Ziel der Nachhaltigkeit im Bereich Software-Entwicklung und Software-Betrieb zu erreichen. In diesem Kapitel werden bereits bestehende Rahmenwerke betrachtet. Es werden aber auch Bereiche angesprochen, in denen noch weitere Handlungsvorschriften hilfreich sein können.

5.1 Woran man sich halten kann

Das Umweltbundesamt hat im Jahr 2020 mehrere Stellungnahme im Umfeld ressourcen- und energieeffizienter IT veröffentlicht, unter anderem zu **↗Rechenzentren** und **↗digitalen Infrastrukturen**. Bitkom hat hierzu bereits **↗Stellung bezogen** und weitere politische Handlungsempfehlungen abgeleitet.

Der Blaue Engel

Eine Möglichkeit, sich in Bezug auf Ressourceneffizienz in einem Rahmen zu bewegen, der vom Gesetzgeber mitgestaltet wird, besteht in der Zertifizierung mit dem Blauen Engel. Der **↗Blaue Engel für »Ressourcen- und energieeffiziente Softwareprodukte« (DE-UZ 215)** ist ein Umweltzeichen, welches unter anderen vom Bundesministerium für Umwelt, Naturschutz und nukleare Sicherheit und dem Umweltbundesamt vergeben wird. Ziel des Zeichens ist die Kennzeichnung von Produkten, »die im besonderen Maße sparsam mit den Hardware Ressourcen umgehen und in ihrer Nutzung einen sparsamen Energieverbrauch aufweisen« (vgl. **↗Vergabekriterien des Blauen Engel für Ressourcen- und Energieeffiziente Softwareprodukte**). Die Kriterien des Blauen Engels für ressourcen- und energieeffiziente Software basiert auf Forschung des **↗Umweltcampus Birkenfeld** (vgl. **↗Projekte im Bereich Green Software Engineering**) und betrachtet in der aktuellen Form Desktop-Computer-Systeme. Eine Erweiterung auf mobile Anwendungen und Anwendungen, die in der Cloud betrieben werden, ist geplant.

Was wird zertifiziert?

Produkte, die hinsichtlich der Eignung bewertet werden sollen, werden in Bezug auf die Minimierung von Ressourcenaufwand und Energiebedarf evaluiert. Insbesondere bezieht sich dies auf:

- Hardware-Auslastung und elektrische Leistungsaufnahme im Leerlauf
- Hardware-Inanspruchnahme und Energiebedarf bei Ausführung eines Standardnutzungsszenarios
- Unterstützung des Energiemanagements

Neben Energie-Effizienz beziehen die Kriterien auch andere Nachhaltigkeits-Faktoren mit ein. Dazu gehört die potentielle Nutzungsdauer von Geräten, die durch Software beeinflusst wird, sowie Nutzungsautonomie. Das umfasst Interoperabilität durch offene Datenformate, Transparenz und Kontinuität des Software-Produkts, Desinstallierbarkeit, Offlinefähigkeit, Modularität, Werbefreiheit, Transparenz der Dokumentation und Nutzungsbedingungen.

Das Umweltzeichen »Blauer Engel« und die damit verbundene positive Marketing-Wirkung dient hier auch als Incentivierung für die Entwicklung nachhaltiger und insbesondere energieeffizienter Software.

5.2 Was noch zu tun ist

Um die Transparenz für **↗Verbraucherinnen und Verbraucher** zu steigern und damit mehr Aufmerksamkeit zu schaffen, sollte Software eine Energieverbrauchskennzeichnung haben. Wie bei anderen Produkten auch, wäre eine ampelartige kategorielle Kennzeichnung denkbar, die in einfacher Art und Weise Auskunft darüber gibt, wie viel Energie eine Software verbrauchen wird.

Ressourcenschonende Softwareentwicklung sollte darüber hinaus gefördert werden. Um Anreize bei den Unternehmen, Startups sowie Entwicklerinnen und Entwicklern zu schaffen und Bewusstsein herzustellen, wäre eine Förderung von Softwareentwicklung, die ressourcenschonend ist, ein geeignetes Mittel. Gleichwohl wäre denkbar, dass Softwareentwicklung gefördert wird, die beim Schonen von Ressourcen und Nachhaltigkeit unterstützt.

Die Anbieter cloudbasierter Services in Europa sollten als Vorreiter im Kampf gegen den Klimawandel fungieren und vor 2030 ihre Energie komplett aus erneuerbaren Ressourcen beziehen. Cloud-Services sind eine Schlüsseltechnologie der nächsten Dekaden und ihr Energieverbrauch wird stark steigen. Gerade deshalb sind sie ein geeigneter Hebel um Ressourceneffizienz zu fördern.

6 Auf einen Blick

Als Abschluss des Leitfadens sind nachfolgend kurz und knapp Empfehlungen für ressourcenschonende, nachhaltige und langlebige Software zusammengefasst, die durch eine Studie des Umweltbundesamtes zusammengetragen wurden (vgl. Gröger und Köhn 2015):

- **Abwärtskompatibilität** – Du solltest Software auch auf Hardware verwenden können die fünf Jahre alt ist (vor bis zu 5 Jahren aktuell war).
- **Auswahl verschiedener Energieeffizienzmodi**: Du solltest Software verwenden können, bei der du selbst über das Verhältnis von Energieverbrauch und Performance entscheiden kannst.
- **Default-Einstellungen**: Du solltest Software verwenden können, deren Energiesparpläne so gestaltet sind, dass nachhaltige Nutzungsmuster voreingestellt sind.
- **Fernwartung**: Du solltest Software anpassen und warten können, ohne dafür gleich die Hardware auszutauschen.
- **Geringe Belastung der Hardware**: Du solltest den Energiebedarf von jeder Hardware messen und mit anderen Hardwareprodukten vergleichen können (unabhängig von der Software).
- **Minimale Datenübertragung über Mobilfunknetze**: Du solltest Software verwenden können die erkennt, wann eine Mobilfunkverbindung besteht, um dann die Datenübertragung zu drosseln oder komplett zu stoppen.
- **Modularität**: Du solltest in Software nur jene Module installieren müssen, die für die jeweilige Anwendung auch wirklich erforderlich sind.
- **Monitoringtools**: Du solltest Software verwenden können, bei der sich der Energieverbrauch, RAM- und CPU-Beanspruchung mit Hilfe von Tools untersuchen lässt, oder sogar direkt in die Benutzeroberfläche der Software integriert ist.
- **Niedriger Energiebedarf**: Du solltest den Energiebedarf von jeder Software messen und mit anderen Softwareprodukten vergleichen können.
- **Offline-Nutzung**: Du solltest Software auch offline nutzen können, mit Hilfe von Export-Schnittstellen und lokalen Kopien der Datenbanken.
- **Off-Schalter**: Du solltest Software verwenden können, bei der du nicht verwendete Hardware-Komponenten abschalten kannst.
- **Plattformunabhängigkeit**: Du solltest Software unabhängig von bestimmter Hardware verwenden können.
- **Schlanke Programmierung**: Du solltest Software verwenden können, die nur genau eine Aufgabe erledigt und diese Aufgabe besonders gut erledigt.

- **Speichereffizienz**: Du solltest Software verwenden können, die nur das Notwendige speichert und automatisiert, die Daten später löscht und sparsame Datenformate verwendet.

- **Transparenz**: Du solltest Software nutzen können, die transparent macht, welche Ressourcen bei der Planung und Implementierung der Software verbraucht wurden.

  - welche Ressourcen die Software benötigt, wenn sie später läuft (RAM, CPU, Netzbelastung, etc.).
  - welche Daten gespeichert werden.
  - wie lange Daten gespeichert werden.
  - welche Daten als Rückstand bei einer Deinstallation verbleiben.

- **Wissensaufbau** – Du solltest Software verwenden, bei der einsehbar und dokumentiert ist, welche Probleme und Lösungen während der Implementierung der Software aufgetaucht sind.

Cheat Sheet

Ressourceneffiziente Programmierung

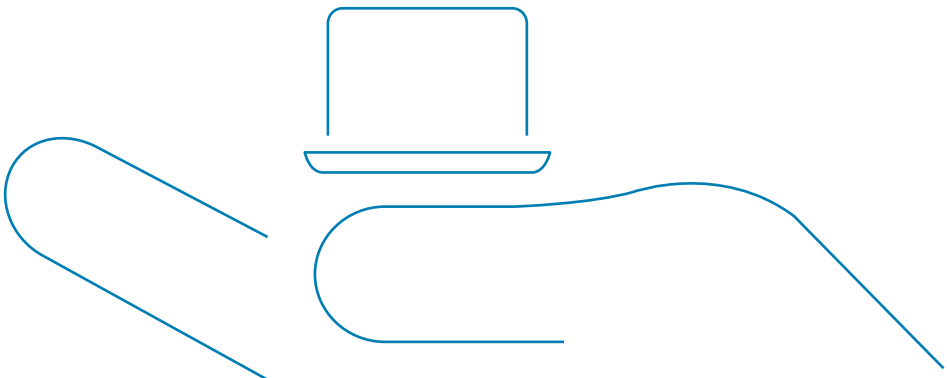
In der aktuellen Entwicklung der Digitalen Transformation ist ein ansteigender Trend zu verzeichnen: Software ist für einen zunehmenden CO2-Ausstoß verantwortlich. Hintergrund dieser Entwicklung ist, dass bisher Nachhaltigkeit und Schonung von Umwelt-Ressourcen bisher nur in geringem Maße bei IT-Projekten berücksichtigt werden.

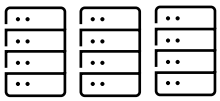
Im Leitfaden »Ressourceneffiziente Programmierung« wird ein Überblick über die Möglichkeiten der Berücksichtigung von Nachhaltigkeit, Langlebigkeit und Ressourceneffizienz bei Softwareentwicklungsprojekten geben. Mit diesem Cheat Sheet wollen ergänzend dazu eine aktive Hilfestellung geben, um verantwortlichen Personen einen Einstieg in das Themenfeld der nachhaltigen und ressourceneffizienzten Softwareentwicklung zu geben.

Autoren dieses Cheat Sheet sind Mitglieder des **↗Arbeitskreises Software Engineering & Software Architektur**. Über Feedback zu diesem Cheat Sheet freuen wir uns sehr. Ansprechpartner ist Dr. Frank Termer (f.termer@bitkom.org).

Serverleistung

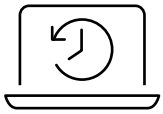
1. Potenzial	2. Messen	3. Umsetzen
Ressourcen einsparen durch Reduzierung der Serverleistung (Suffizienz)	▪ Server-Leistung ▪ Hostingkosten *Leistung korreliert grundsätzlich mit Stromverbrauch, CO2-Emissionen und Kosten.*	▪ Serverleistung (Prozessoren, RAM, Netzwerk, Speicher) an Bedarf orientieren ▪ keine übermäßigen Leistungswerte beschaffen
Ressourcen einsparen durch bedarfsgerechte Skalierung (Suffizienz)	▪ Skalierbarkeit der einzelnen Komponenten ▪ Anzahl tatsächlich durchgeführter Up- & Down-Skalierungen *Durch Skalierung individueller Komponenten soll der Gesamtressourcenverbrauch reduziert werden.*	▪ Modularisierung und Entkopplung von Komponenten ▪ bei Bedarf nur die Komponenten skalieren, die nötig sind ▪ automatisierte, lastabhängige Skalierung umsetzen





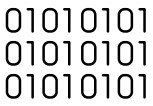
Rechenzentren

1. Potenzial	2. Messen	3. Umsetzen
Ressourcen einsparen durch Nutzung erneuerbarer Energien	Anteil erneuerbarer Energien *Durch den Einsatz erneuerbarer Energien kann ein geringerer Verbrauch von Umwelt-Ressourcen erreicht werden.*	Rechenzentren mit erneuerbarer Energieversorung präferieren
Ressourcen Einsparen durch die Nutzung energieeffizienter Rechenzentren	▪ Zertifizierung des Rechzentrums, z.B. der blaue Engel ▪ Power-Usage-Effectiveness-Koeffizent des Rechenzentrums *Durch Skalisierungseffekte sind moderne, hochspezialisierte Rechenzentren hocheffizient – meist effizienter als On-Premise-Lösungen.*	▪ Moderne Rechenzentren nutzen ▪ On-Prem-Umgebungen ggf. migrieren



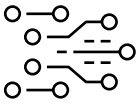
Obsoleszenz

1. Potenzial	2. Messen	3. Umsetzen
Ressourcen einsparen durch längere Nutzung von Hardware	▪ Nutzungsdauer der Hardware ▪ Updatebarkeit des Systems, um auf geänderte Anforderungen reagieren zu können *Durch die längere Nutzung von Hardware können Produktionsressourcen geschont sowie Elektroschrott vermieden werden.*	▪ Existierende Hardware länger nutzen – so lange sinnvoll ▪ Vor neuer Beschaffung die Abwärtskompatibilität, Modularität und Austauschbarkeit von Hardwarekomponenten prüfen



Software

1. Potenzial	2. Messen	3. Umsetzen
Ressourcen einsparen durch die Verwendung energieeffizienter Programmiersprachen	▪ Prozesslaufzeit ▪ CPU-Last ▪ Speicherverbrauch *Durch die Nutzung effizienter Sprachen und Programmierung kann ein geringerer Verbrauch von Hardwareresourcen ermöglicht werden (→ Serverleistung).*	▪ Vergleiche bzgl. des Energie- oder Ressourcenvebrauchs von Programmiersprachen berücksichtigen ▪ effiziente Sprachen nutzen
Ressourcen einsparen durch effiziente Programmierung innerhalb einer Sprache		Effizienten und schlanken Code programmieren, z.B. geeignete Datenstrukturen, Iterationen durch Vektorisierung statt Schleifen



Architektur

1. Potenzial	2. Messen	3. Umsetzen
Ressourcen einsparen durch eine bedarfsgerechte Architektur	▪ Menge des Datenverkehrs ▪ Gesamtverbrauch von Ressourcen auf Servern und Clients *Durch Einsparen von Berechungen können Datenverkehr und Server-Ressourcen eingespart werden.*	▪ Komplexe Berechnungen dort durchführen, wo sie am effizientesten sind ▪ Datenlokalität berücksichtigen, Berechnungen ggf. dezentralisieren

Anforderungsmanagement

1. Potenzial	2. Messen	3. Umsetzen
Ressourcen einsparen durch Aufnahme von Nachhaltigkeitskriterien als Anforderungen	▪ Nachhaltigkeitskriterien als Anforderungen ▪ Formulierte Zielstellungen bzgl. Nachhaltigkeit *Durch das Bewusstsein und Verfolgen von Nachhaltigkeitskriterien werden meist erfolgreich Ressourceneinsparungen umgesetzt.*	▪ Nutzung etablierter nicht-funktionaler Anforderungen (z.B. Datensparamkeit, Offene Schnittstellen, Performance) ▪ Bezug etablierter nicht-funktionaler Anforderungen zur Nachhaltigkeit herstellen ▪ Nutzung von Frameworks wie »Sustainability for non-functional-requirements«

Transparenz

1. Potenzial	2. Messen	3. Umsetzen
Ressourcen einsparen durch Bewusstsein für deren Verbrauch	▪ Informationen über Ressourcenverbräuche sind bekannt ▪ Entscheidungsträger unterstützen das Bewusstsein für Ressourcenverbräuche ▪ Incentivierung für ressourcenschonendes Vorgehen ist vorhanden *Durch ein unternehmensweites Bewusstsein für ressourcenschonendes Handeln, kann eine Breitenwirkung von Maßnahmen erreicht werden.*	▪ Ressourcenverbrauch für alle Stakeholder transparent machen, z.B. anhand von KPIs, Ampeln und Zertifikaten ▪ Nachhaltigkeit bei der Incentivierung berücksichtigen

Literaturverzeichnis

Amsel, N.;, Ibrahim, Z.; Malik, A.; Tomlinson, B. (2011) Toward sustainable software engineering. In Proceedings of the 33rd International Conference on Software Engineering (ICSE '11). Association for Computing Machinery, New York, NY, USA, 976–979
↗**https://doi.org/10.1145/1985793.1985964**
abgerufen am 15.02.2021

Andrae, A.; Edler, T. (2015) On Global Electricity Usage of Communication Technology: Trends to 2030, Challenges, 6(1), 117–157
↗**https://doi.org/10.3390/challe6010117**
abgerufen am 15.02.2021

Ardito, L.; Procaccianti, G.; Torchiano, M.; Vetro, A. (2015) Understanding Green Software Development: A Conceptual Framework. IT Professional, vol. 17, no. 1, pp. 44-50, doi
↗**https://doi.org/10.1109/MITP.2015.16**
abgerufen am 15.02.2021

Bitkom (2020) Klimaschutz durch digitale Technologien – Chancen und Risiken.
↗**https://www.bitkom.org/klimaschutz-digital**
abgerufen am 15.02.2021

Bonér, J.; Farley, D.; Kuhn, R.; Thompson, M. (2014) Das Reaktive Manifest.
↗**https://www.reactivemanifesto.org/de**
abgerufen am 15.02.2021

Cassel, D. (2018) Which Programming Languages Use the Least Electricity?
↗**https://thenewstack.io/which-programming-languages-use-the-least-electricity/**
abgerufen am 15.02.2021

Gröger,J.; Köhn, M. (2015) Nachhaltige Software – Dokumentation des Fachgesprächs »Nachhaltige Software« am 28.11.2014
↗**https://www.umweltbundesamt.de/publikationen/nachhaltige-software**
abgerufen am 15.02.2021

Hauff, V. (Hrsg.) (1987) Unsere gemeinsame Zukunft. Der Brundtland-Bericht der Weltkommission für Umwelt und Entwicklung. Eggenkamp, Greven.

Jones, N. (2018) How to stop data centres from gobbling up the world's electricity. Nature 561, 163-166.
↗**https://doi.org/10.1038/d41586-018-06610-y**
abgerufen am 15.02.2021

Lago, P. (2018) Green Software: Architecture Decision-making for Sustainability.
↗**https://de2.slideshare.net/patricia_lago/green-software-architecture-decisionmaking-for-sustainability**
abgerufen am 15.02.2021

Naumann, S.; Dick, M.; Kern, E.; Johann, J. (2011) The GREENSOFT Model: A reference model for green and sustainable software and its engineering, Sustainable Computing: Informatics and Systems, 294-304. ↗**https://doi.org/10.1016/j.suscom.2011.06.004**
abgerufen am 15.02.2021

Pereira, R.; Couto, M.; Ribeiro, F.; Rua, R.; Cunha, J.; Fernandes, J.; Saraiva, J. (2017) Energy efficiency across programming languages: how do energy, time, and memory relate? In Proceedings of the 10th ACM SIGPLAN International Conference on Software Language Engineering (SLE 2017). Association for Computing Machinery, New York, NY, USA, 256–267
↗**https://doi.org/10.1145/3136014.3136031**
abgerufen am 15.02.2021

Raturi, A.; Penzenstadler, B.; Tomlinson, B.; Richardson, D. (2014) Developing a sustainability non-functional requirements framework. In Proceedings of the 3rd International Workshop on Green and Sustainable Software (GREENS 2014). Association for Computing Machinery, New York, NY, USA, 1–8.
↗**https://doi.org/10.1145/2593743.2593744**
abgerufen am 15.02.2021

Singh, K. (2020) Spring Webflux: EventLoop vs Thread Per Request Model.
↗**https://dzone.com/articles/spring-webflux-eventloop-vs-thread-per-request-mod**
abgerufen am 15.02.2021

St. Laurent, J. (2018) The Art of Sizing Your IT Infrastructure.
↗**https://www.infoworld.com/article/3261604/the-art-of-sizing-your-it-infrastructure.html**
abgerufen am 15.02.2021

The Open Group (2018) The TOGAF® Standard, Version 9.2.
↗**https://pubs.opengroup.org/architecture/togaf9-doc/arch/chap20.html**
abgerufen am 15.02.2021

Vereinte Nationen (Hrsg.) (2016) Ziele für nachhaltige Entwicklung.
↗**https://unric.org/de/17ziele/**
abgerufen am 15.02.2021

Bitkom vertritt mehr als 2.700 Unternehmen der digitalen Wirtschaft, davon gut 1.900 Direktmitglieder. Sie erzielen allein mit IT- und Telekommunikationsleistungen jährlich Umsätze von 190 Milliarden Euro, darunter Exporte in Höhe von 50 Milliarden Euro. Die Bitkom-Mitglieder beschäftigen in Deutschland mehr als 2 Millionen Mitarbeiterinnen und Mitarbeiter. Zu den Mitgliedern zählen mehr als 1.000 Mittelständler, über 500 Startups und nahezu alle Global Player. Sie bieten Software, IT-Services, Telekommunikations- oder Internetdienste an, stellen Geräte und Bauteile her, sind im Bereich der digitalen Medien tätig oder in anderer Weise Teil der digitalen Wirtschaft. 80 Prozent der Unternehmen haben ihren Hauptsitz in Deutschland, jeweils 8 Prozent kommen aus Europa und den USA, 4 Prozent aus anderen Regionen. Bitkom fördert und treibt die digitale Transformation der deutschen Wirtschaft und setzt sich für eine breite gesellschaftliche Teilhabe an den digitalen Entwicklungen ein. Ziel ist es, Deutschland zu einem weltweit führenden Digitalstandort zu machen.

Bundesverband Informationswirtschaft,
Telekommunikation und neue Medien e.V.

Albrechtstraße 10
10117 Berlin
T 030 27576-0
F 030 27576-400
bitkom@bitkom.org
www.bitkom.org

bitkom