Grüne Software-Engineering Techniken zur Entwicklung energieeffizienter Software

Aleksandr Chmelev, Fatima Zahra En-Nasiry, Nargiz Sadykova

UV Wissenschaftliche Arbeitstechniken und Präsentation | Wintersemester 25/26

FB Informatik

Universität Salzburg

Motivation:

- Energieverbrauch steigt
- Software beeinflusst Energie und CO_2
- Rechenzentren/Cloud
- Kostenersparnis
- Verantwortung

Inhalt:

- Green Software Engineering
- Energieeffiziente Software
- Methoden zur Entwicklung energieeffizienter Software
- Schlüsselkomponente
- Anwendungsbeispiele
- Fazit

Was bedeutet "*grün*" im *Software Engineering?*

Entwicklung von Software mit dem Ziel, Energieverbrauch und Umweltwirkungen durch softwarebezogene Entscheidungen zu minimieren

- Energieeinsparung durch Architektur-, Design- und Codeentscheidungen

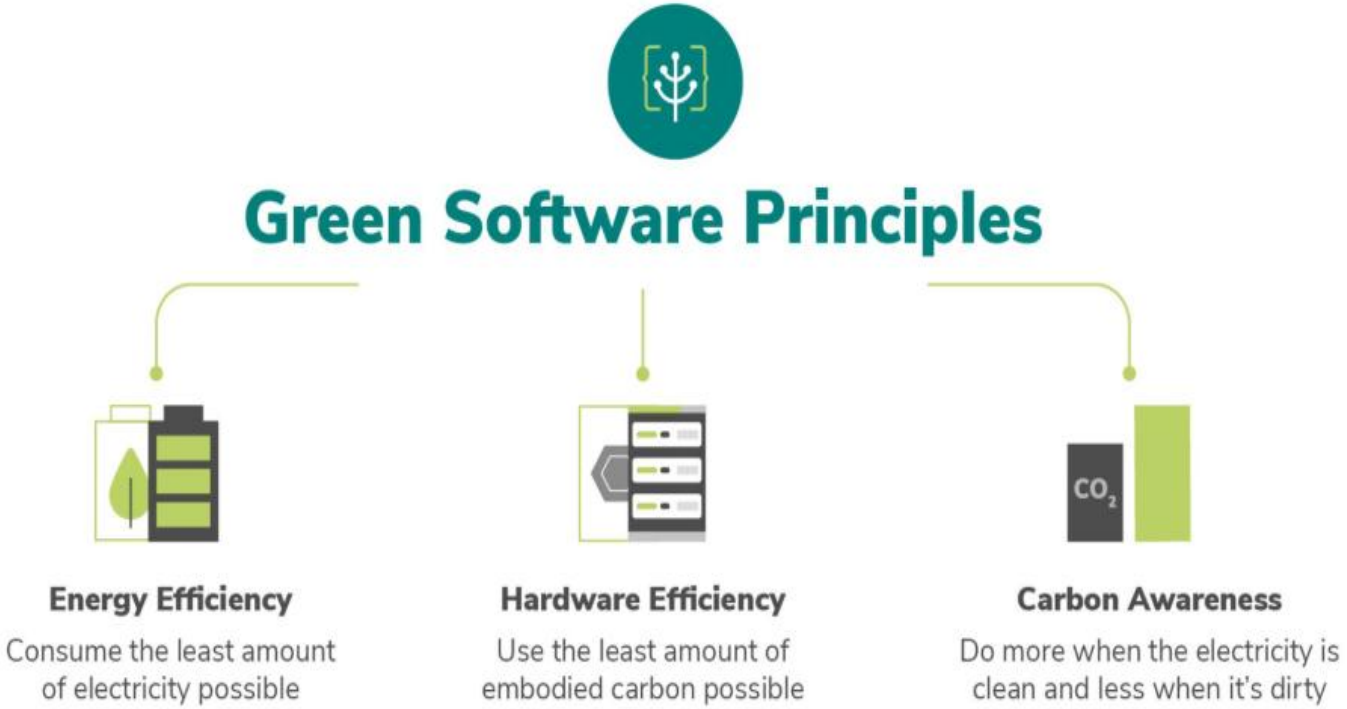
- Berücksichtigung von Entwicklung, Betrieb und Nutzung

- Nicht Hardware, sondern Software bestimmt Ressourcennutzung

Ziele von Green Software Engineering:

- Energieeffizienz
- Verringerungen von CO_2-

  Emissionen
- Hardwareeffizienz



Energieeffiziente Software (I)

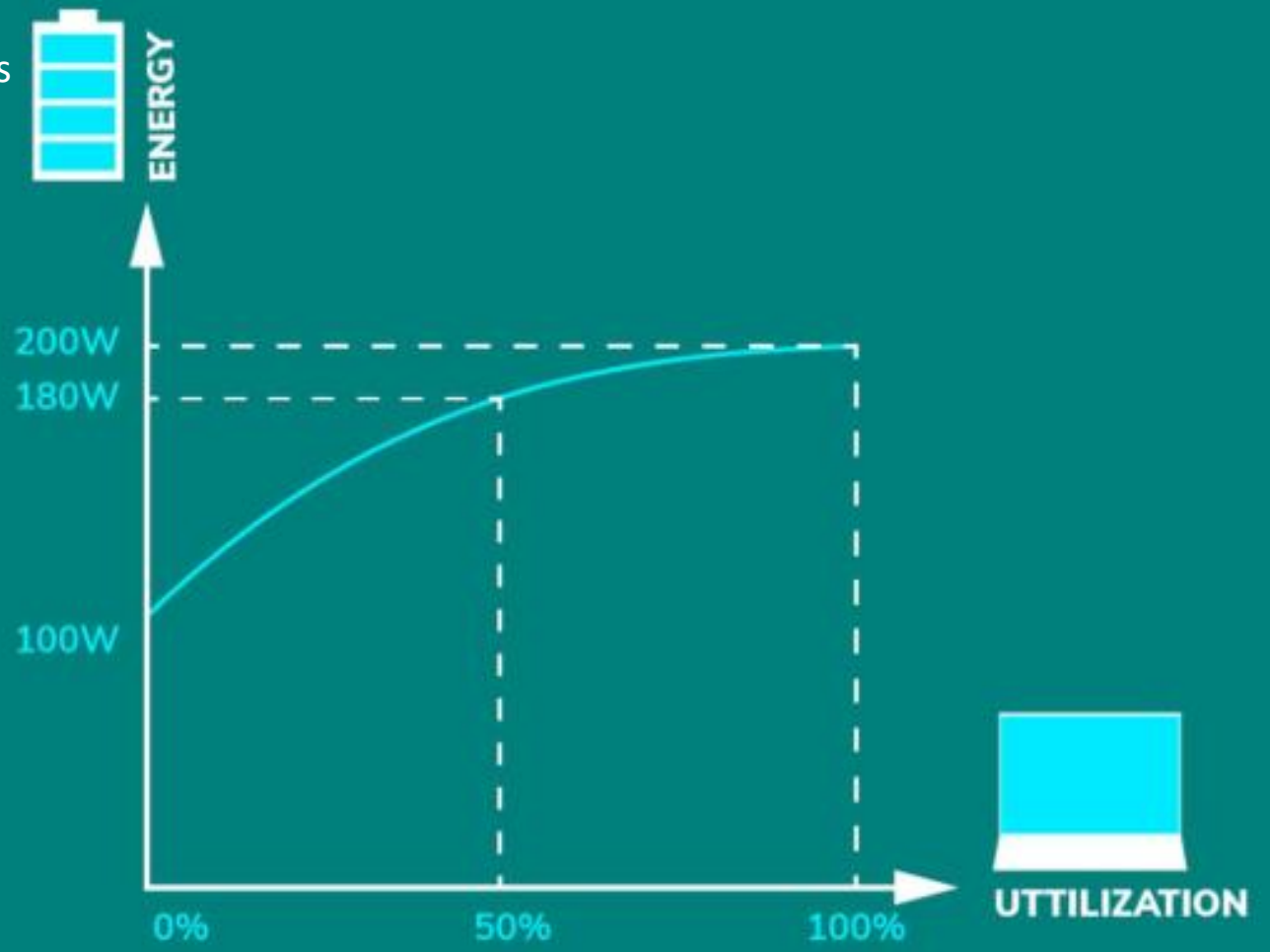
Das Ziel ist, Software so zu entwickeln, dass dieselbe Aufgabe mit weniger Rechenzyklen und geringerem Energieverbrauch ausführt wird

Weniger Berechnungen ⟶ Weniger Energieverbrauch

Energieeffiziente Software (II)

- Hoher Grundenergieverbrauch bereits bei geringer Auslastung
- Energie steigt nicht proportional zur Nutzung
- Leerlauf vermeiden, Auslastung optimieren

Energieverbrauch eines Systems



Systemauslastung

Green Software Foundation
greensoftware.org

creative commons

Verringerung von CO2-Emissionen

- CO_2-Emissionen hängen von Zeit und Ort des Energieverbrauchs ab
- Stromnetze haben unterschiedliche CO_2-Intensität
- Green SWE nutzt Energie, wenn sie klimafreundlicher ist

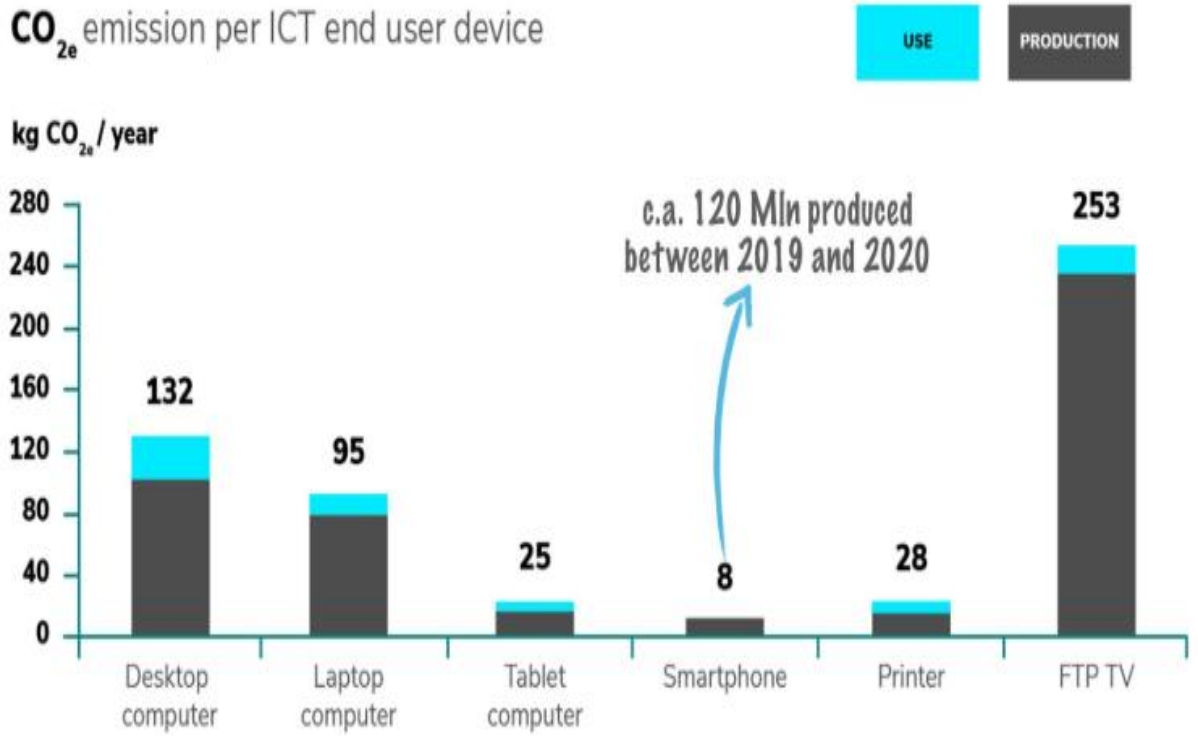
Hardwareeffizienz (I)

- Reduktion von „Embodies Carbon“
- Längere Nutzung bestehender Hardware
- Weniger Neuproduktion durch effiziente Software

⟶ Gute Software verlängert das Leben der Hardware und spart CO_2

Hardwareeffizienz (II):

- Ein Großteil der CO_2-Emissionen entsteht bei der Hardwareproduktion
- Nutzung verursacht oft weniger Emissionen als Herstellung
- Effiziente Software verlängert die Hardware - Lebensdauer



Green Software Foundation
greensoftware.org

Entwicklung energieeffizienter Software

- Qualitätsmerkmal

- Ziel in der Entwicklung

Energieeffiziente Software als Ziel in der Entwicklung

- Algorithmen und Datenstrukturen
- Frameworks
- Architekturmuster

Algorithmische Optimierung

- Vermeidung redundanter Berechnungen
- Effiziente Algorithmen und Datenstrukturen
- Weniger Rechenoperationen ⟶ weniger Energie

Frameworks

- Vorgefertigte optimierte Software-Bausteine
- Reduzieren unnötige Hintergrundprozesse
- Effiziente Nutzung von CPU und Speicher
- Erleichtern wartbare & ressourcenschonende Entwicklung

Energieeffiziente Softwarearchitektur

- Designentscheidungen steuern Energieverbrauch
- Ressourcen nur bei Bedarf nutzen (Auto-Skalierung)
- Weniger Datenübertragung durch Caching
- Architektur bestimmt langfristigen Energieverbrauch

Schlüsselkomponente

- Analyse konkreter Software
- Priorisierung der Ziele
- Optimale Entscheidungen in frühen Entwicklungsphasen

Anwendungsbeispiele:

Algorithmische Optimierung – Sortieralgorithmen

Vom klassischen Quicksort zum Dual-Pivot Quicksort:

Reduzierte Anzahl von Speicherzugriffen

- Geringere Anzahl von Vergleichen
- >10% reduzierte Laufzeit ⟶ geringerer Energieverbrauch

Frameworks

Quarkus

Java-Framework für die Entwicklung von Cloud- und Microservice-Anwendungen

- Fokus auf geringere Ressourcennutzung
- Reduzierter Laufzeit-Overhead durch Build-Time-Optimierungen
- Schneller Start und Geringer Speicherbedarf

Architekturmuster

Mikroservice-Architektur

Aufteilung einer Anwendung in mehrere eigenständige Services mit klar definierten Schnittstellen und Verantwortlichkeiten

- Energieverbrauch ist stark kontextabhängig
- Anhängig von Antwendung und Umsetzung können Energiekosten um entweder 20-40% steigen oder um 15-20% reduziert werden

Fazit

- Green Software Engineering betrachtet die Softwareentwicklung in ihrer Gesamtheit

- Energieeffizienz ist ein Ergebnis vieler miteinander verbundener Entscheidungen während des gesamten Software-Lebenszyklus

- Eine universelle Standardlösung existiert nicht

Quellen:

Ardito, L., Procaccianti, G., Torchiano, M., & Vetrò, A. (2015).
Understanding Green Software Development: A Conceptual Framework. IEEE Software, 32(1), 35–44.

Danushi, A., Procaccianti, G., Lago, P., & Auer, F. (2024).
Architectural Design Decisions for Energy-Efficient Software Systems: A Systematic Review. Journal of Systems and Software, 208, 111824.

de Nardin, I. F., Righi, R. da R., Lopes, T. R. L., da Costa, C. A., Yeom, H. Y., & Köstler, H.
On Revisiting Energy and Performance in Microservices Applications: A Cloud Elasticity-Driven Approach.

Eder, K., Gallagher, J., López-García, P., et al. (2016).
ENTRA: Whole-Systems Energy Transparency.

Fegno Technologies. (o. J.).
Energy Efficiency in Software Systems.

Field, T., Anderson, S., & Eder, K. (2014).
Making Energy Consumption Data Accessible to Software Developers.

Green Software Foundation. (2025).
Green Software Engineering Principles.

Hilty, L. M., & Aebischer, B. (2015).
ICT for Sustainability: An Emerging Research Field. In *ICT Innovations for Sustainability*. Springer, Cham.

International Journal for Research Trends and Innovation.
Energy-Efficient Algorithms for Sustainable Software Development.

Methodda. (o. J.).
Energy Efficiency-Focused Software Development Strategies.

Sallou, M., Lague, S., et al. (2023).
EnergiBridge: Empowering Software Sustainability through Cross-Platform Energy Measurement.

Wild, S., Nebel, M. E., & Martínez, C.
Analysis of Pivot Sampling in Dual-Pivot Quicksort.