Nachhaltige IT-Infrastrukturen – Ansätze für ressourcenbewusste Softwareentwicklung: SSH & Vim & VSCodium

Autor:innen:
Studierende Informatik: Franjo Gießel, Laura Halbeck, Julian Nehring, Silja Rehmke, Jan Ole Seutter, Alexey Spitsyn, Lennart Steffen, Maxim Ziegler
Studierende Wirtschaftsinformatik: Lamis Aiche
Betreuung: Oliver Radfelder

Im Rahmen der FIfF-Konferenz 2024 haben wir, eine Studierendengruppe aus den Bachelorstudiengängen Informatik und Wirtschaftsinformatik der Hochschule Bremerhaven, im Workshop „Ressourcenbewusste Softwareentwicklung: SSH & Vim & VSCodium“ unser Bachelor-Projekt vorgestellt, das sich mit der nachhaltigen Nutzung von IT-Ressourcen befasst. Ziel des Workshops war es, praktische Ansätze zur Ermittlung und Minimierung des Ressourcenverbrauchs in einer Testinfrastruktur aufzuzeigen und gemeinsam mit den Teilnehmenden zu erproben und zu diskutieren.

Hintergrund

In den Informatikstudiengängen in Bremerhaven werden Studierende von Beginn an in eine eigene *GNU/Linux*-Server-Infrastruktur eingeführt. Diese Infrastruktur ist über das Terminal und SSH für alle Studierenden zugänglich und bietet einen fertig konfigurierten sowie gemeinsam nutzbaren Unix-Workspace. Das Ziel besteht darin, eine offene, freie und niedrigschwellige Arbeits- und Programmierumgebung für alle zu schaffen, die die folgenden Anforderungen erfüllt:

1. Die Infrastruktur soll mit älteren oder kostengünstigen Geräten nutzbar sein, um Zugangshürden zu minimieren.
2. Sie muss plattformunabhängig sein, da Studierende unterschiedliche Endgeräte und Betriebssysteme verwenden und niemandem ein spezifisches System vorgeschrieben werden soll.
3. Zuverlässigkeit ist essenziell, insbesondere bei der Nutzung im Remote-Betrieb für die Lehre.

4. *Open Source* soll die Freiheit in der Lehre unterstützen und praktische Hürden durch unübersichtliche Lizenzbedingungen vermeiden.
5. Nachhaltigkeit spielt eine entscheidende Rolle, da der ökologische Fußabdruck von IT-Infrastrukturen reduziert werden soll und unsere Infrastruktur eine Vorbildfunktion einnehmen sollte.

Diese Anforderungen bilden die Grundlage für die Umsetzung ressourcenbewusster IT-Infrastrukturen, die sowohl für Bildungszwecke als auch für den praktischen Einsatz geeignet sind. Sie umfassen nicht nur technische, sondern auch soziale Aspekte. So erfordert die Gestaltung einer solchen Infrastruktur eine enge Zusammenarbeit zwischen Studierenden und Lehrenden, um sicherzustellen, dass alle Bedürfnisse berücksichtigt werden. Die Bereitstellung einer offenen Plattform fördert nicht nur die technische Bildung, sondern ermöglicht auch die Entwicklung kritischen Denkens im Umgang mit digitalen Ressourcen. Entscheidend ist, dass nachhaltige IT-Lösungen als praxisorientierte Lernobjekte dienen können, um technische Kompetenz mit ökologischer Verantwortung zu verbinden.

Das Bachelor-Projekt hat sich zum Ziel gesetzt, eine Testinfrastruktur auf den Ressourcenverbrauch in einem typischen Nutzungsszenario zu untersuchen und systematisch zu hinterfragen. Für diesen Zweck haben wir einen, der Informatik-Infrastruktur nachgebildeten, Remote-Workspace aufgebaut und führen automatisierte Tests durch, in denen ein vollständiger Entwicklungsprozess (Programmieren, Kompilieren, Testen, Deployment) simuliert wird.

Hier setzte der Workshop an, in dem die Teilnehmer:innen die Möglichkeit hatten, in einer Live-Demo und in praktischen Hands-On-Experimenten das Projekt-Setup zu erproben. Im Anschluss wurden die Zwischenergebnisse des Experimentier-Setups und weitere Aspekte einer nachhaltigen Infrastruktur diskutiert. Der Workshop setzte einen Schwerpunkt auf die Nutzung von Open Source Tools (*Vim* und *VSCodium*) und die Diskussion, wie gebrauchte und refurbished Hardware eingesetzt werden kann, um eine möglichst nachhaltige, effiziente und zugleich leistungsfähige Entwicklungsumgebung zu schaffen.

Das Projekt

Das Bachelor-Projekt *Konsequent Open Source basierte und nachhaltige Entwicklungsumgebung* widmet sich der Frage, wie IT-Infrastrukturen ressourcenschonend gestaltet und genutzt werden können, insbesondere im Kontext der akademischen Lehre. Ziel des Projekts ist, durch eine Kombination aus empirischen Messungen und theoretischen Analysen ein besseres Verständnis für die Energieeffizienz zentralisierter und

dezentralisierter IT-Lösungen zu gewinnen. Wir beziehen uns dabei auf das in einer Studie des Fraunhofer ISI genutzte Modell der drei Effektebenen der Digitalisierung hinsichtlich Umwelt- und Klimawirkungen (vgl. Köppl-Turyna et. al. 2021). Dieses Modell unterscheidet zwischen der Technologie- (Lebenszyklus), der Mikro- (Anwendungen) und der Makro-Ebene (Reboundeffekt, Struktur und Verhaltensänderungen).
In unserem Projekt untersuchen wir auf der Mikroebene die Last und den Energiebedarf von End- und Servergeräten in konkreten Nutzungsszenarien (lokal und remote) sowie die Bewertung und Effizienz gängiger Software-Werkzeuge (Vim und VSCodium). Auf der Technologie-Ebene ging es darum, mit einer Literaturrecherche die CO2e-Emissionen (CO2-äquivalenten-Emissionen) von Hardware über den gesamten Lebenszyklus besser zu verstehen und zu überprüfen, ob sich 1. der Einsatz gebrauchter Hardware günstig auf die CO2e-Emissionen auswirkt und 2., ob geringere technische Anforderungen an Endgeräte deren Lebensdauer erheblich verlängern können. Reboundeffekte treten bei technischen Effizenzgewinnen regelmäßig auf und erfordern strukturelle und Verhaltensänderungen, um systematisch adressiert zu werden.

[Abbildung 1; Bildunterschrift: *Nachhaltigkeit und Effektebenen der Digitalisierung (in Anlehnung an Köppl-Turyna et. al. 2021: S. 11; Horner et al. 2016; Berkhout & Hertin 2004; Hilty et al. 2006; Williams 2011; Rattle 2010)*]

Aufbau der Testinfrastruktur und Methodik

Im Rahmen des Studiums erfolgt die Einführung in den Umgang mit dem Terminal-Editor *Vim* sowie *VSCodium*, einem browserbasierten, erweiterbaren Open-Source-Texteditor, der u.A. in GitHub-Codespaces oder Gitlab verwendet wird.
Typischerweise verbinden sich Studierende im Terminal per SSH mit dem Workspace der Hochschul-Infrastruktur, um hier Projekte zu entwickeln. Dort stehen jedem Studierenden u.a. ein Container mit eigenem Apache- und Tomcat-Server, einer Verbindung zu einem zentralen Datenbankserver und eine große Auswahl an Software-Tools, Programmiersprachen und Compilern zur Verfügung.
Alternativ zum Terminal kann VSCodium mit der *Open Remote SSH*-Extension verwendet werden. Die Extension baut ebenfalls per SSH eine Verbindung zum Remote-Workspace auf. Dort wird zunächst ein VSCodium-Server heruntergeladen und installiert. Auch hier kann dann der vollständige Entwicklungsprozess auf dem Workspace-Server erfolgen.
Durch anwendungsspezifische Erweiterungen kann VSCodium von einem Texteditor in Anwendung und Funktionsumfang vergleichbar zu einer vollständigen IDE aufgewertet werden. Hierzu gehört vor allem die Verwendung eines Language-Servers. In der

Testdurchführung wird zwischen einem VSCodium ohne und mit Erweiterung unterschieden, ersteres wird als *VSCodium-Light*, letzteres als *VSCodium-Full* bezeichnet.

[Abbildung 2; Bildunterschrift: *Vereinfachte Darstellung der Informatik-Infrastruktur an der Hochschule Bremerhaven*]

Damit wir miteinander vergleichbare Tests durchführen und genügend Last auf einem Server erzeugen können, um signifikant messbare Ergebnisse zu ermitteln, haben wir den Entwicklungsprozess eines Studierenden, der eine *Java-Web-App* programmiert, simuliert. Dieser Test soll dann in bis zu 100 Containern gleichzeitig durchgeführt werden.
Für die Simulation wurde für das Terminal ein Python-Skript programmiert, das über einen Parser auszuführende Befehle einliest und dann Tastenanschläge und Interaktionen im Terminal simuliert. Für VSCodium wurde eine entsprechende Extension entwickelt, die ebenfalls mit einem Parser dieselbe zugrunde liegende Web-App einliest, den Editor direkt steuert und dieselben Aktionen durchführt.
Die Simulation erfolgt in einem *Automation-Container*, von wo aus die Entwicklung der Java-Web-App in einem *Workspace-Container* gesteuert wird. Die Workspace-Container befinden sich auf dem zu testenden *System-under-Test* (SuT), je nach Testfall, einem eigenen Server oder einem Laptop.
Damit unterschiedliche Plattformen und Systeme miteinander verglichen werden können, wird als Leistungsindikator des SuT die (erhöhte) Leistungsaufnahme in Watt gemessen, die in etwa proportional zur CPU-Auslastung steigt. Als Messinstrument dient eine *Smart-Steckdose* (Shelly), für die eine Messungenaugigkeit von etwa 2% ermittelt wurde.

[Abbildung 3; Bildunterschrift: *Vereinfachte Darstellung der Test-Infrastruktur*]

Testfälle

Zur Evaluierung der Energieeffizienz werden unterschiedliche Nutzungsszenarien definiert und getestet. Die Fragestellungen umfassen einen Vergleich der Leistungsaufnahme zwischen lokalem und serverbasiertem Arbeiten, die Untersuchung der Effizienz gängiger Software-Werkzeuge wie Vim und VSCodium und (als Beispiel einer für VSCodium typischen Extension) die Bewertung des Ressourcenbedarfs von Language-Servern, die durch Funktionen wie Auto-Vervollständigung und Code-Analyse den Entwicklungsprozess unterstützen.

Die Tests werden in je zwei Konfigurationen durchgeführt:

1. Laptop-basierte Tests: Die Leistungsaufnahme eines typischen Endgeräts wird gemessen, während Aufgaben entweder lokal oder remote auf dem Server ausgeführt werden.
2. Server-basierte Tests: Der Fokus liegt auf der Analyse des Ressourcenverbrauchs des Workspace-Servers, insbesondere bei der Verarbeitung von Aufgaben, die gleichzeitig von mehreren Nutzenden ausgeführt werden.

[Abbildung 4; Bildunterschrift: *Testaufbau der Laptopmessung*]
[Abbildung 5; Bildunterschrift: *Testaufbau der Servermessung*]

Daraus wurden die folgenden Testfälle abgeleite (vgl. Tabelle 1)t:

[Tabelle 1; Bildunterschrift: *Testfälle*]

Für die Tests werden Geräte mit folgenden Spezifikationen genutzt:

- Workspace-Server: Intel Xeon Silver 4214 @ 12x2.20 GHz, 64 GB RAM, 2x 500 GB HDD
- Laptop: Intel Core i5-6300U @ 2x2.40 GHz, 8 GB RAM, 256 GB SSD

Im ersten Aufbau (vgl. Abbildung 4) wird die Leistungsaufnahme des Laptops gemessen, dieser ist an ein Messgerät angeschlossen. Eine direkte Verbindung zum Workspace-Server vom Laptop wird nur in den Testfällen 4 bis 6 hergestellt. Im zweiten Aufbau (vgl. Abbildung 5) wird der an das Messgerät angeschlossene Workspace-Server gemessen.

Erste Ergebnisse

Die ersten Messungen zeigen, dass Vim in allen Testfällen weniger Leistung benötigt als VSCodium-Light (etwa 18%). Besonders auffällig ist jedoch der höhere Ressourcenbedarf von VSCodium-Full (also mit Language-Server), mit im Durchschnitt etwa 25% höherer Leistungsaufnahme gegenüber VSCodium-Light. Auch bei der Analyse der Grafen fällt auf, dass die Last auf dem Workspace-Server deutlich ansteigt, sobald Java-Dateien geöffnet und bearbeitet werden.

In bestimmten Testfällen zeigte sich, dass die Leistungsaufnahme des Laptops im Remote-Betrieb niedriger war als im lokalen Betrieb. Der Betrieb eines zentralen Workspace-Servers verursacht pro Container eine zusätzliche, durchschnittliche Leistungsaufnahme von 3W, während auf dem Laptop durch das Auslagern des Workspaces etwa 2,5W eingespart werden. Die Grundleistungsaufnahme des Servers beträgt 63W, unabhängig von der Anzahl der aktiven Container.

[Abbildung 6; *Leistungsaufnahme des Workspace-Servers bei 16 Containern mit VSCodium Full (Fall 3)*]

Die Untersuchungen zeigen Indizien dafür, dass serverbasierte Entwicklungsumgebungen im Vergleich zu lokalen Setups eine potenziell nachhaltigere Alternative darstellen. Während die Leistungsaufnahme auf dem Laptop durch das Verschieben der Rechenlast auf den Server reduziert wird, bleibt die Gesamtleistungsaufnahme nahezu konstant. Die Ergebnisse deuten darauf hin, dass insbesondere Vim energieeffizienter als VSCodium ist, vor allem aber, dass der Einsatz von Language-Servern bei der Entwicklung einen signifikanten Unterschied ausmacht.
Einschränkend ist zu erwähnen, dass die Ergebnisse aufgrund der begrenzten Anzahl an Testläufen und simulierten Nutzer:innen nicht abschließend sind. Zudem bleibt der Einfluss der Netzwerkverbindung bisher unberücksichtigt. Zukünftige Untersuchungen sollen eine größere Stichprobe und eine höhere Nutzer:innenzahl einbeziehen, um genauere Rückschlüsse zu ermöglichen.
Langfristig könnten serverbasierte Arbeitsweisen dazu beitragen, die Lebensdauer von Endgeräten zu verlängern, da diese weniger leistungsstarke Hardware benötigen. Zudem können Server durch geteilte Ressourcen (CPU, RAM, etc.) effizienter eingesetzt werden.

[Abbildung 7; Bildunterschrift: *Durchschnittlicher Energiebedarf in Watt*
(oben Testfall 1-3, unten Testfall 4-9)]

Nachhaltiges Handeln in der Informatik

Im Zusammenhang des Projekts und im Anschluss an den Workshop haben wir uns mit weiteren Nachhaltigkeitsaspekten befasst, die im Folgenden diskutiert werden. Der Open-Source-Gedanke war von Beginn an zentraler Gegenstand des Projekts. Insbesondere inspiriert durch die Diskussionen im Rahmen der FIfF-Kon 2024 sowie die Reflexionen rund um den Workshop, haben wir uns zudem intensiv mit dem Thema *Green Coding*

auseinandergesetzt.
In der Informatik-Infrastruktur wird, wenn möglich und sinnvoll, gebrauchte und refurbished Hardware eingesetzt. Um über diesen Aspekt fundierte Aussagen treffen zu können, haben wir uns in einer ausführlichen Literaturrecherche mit dem Stand der wissenschaftlichen Forschung zum Thema refurbished und gebrauchter Hardware im Endgerät- und Server-Bereich befasst.

Open Source und FLOSS

Es gibt zahlreiche Gründe, insbesondere im Hochschulkontext, ausschließlich auf Open-Source-Technologie zu setzen. Für das vorliegende Projekt ist Open Source im Wesentlichen ein Mittel, um nachhaltige Konzepte zu realisieren.
Wir verstehen Nachhaltigkeit als ein vielschichtiges, mehrdimensionales Konzept. Daher diskutieren wir Nachhaltigkeit und Open-Source nicht exklusiv unter ökologischen, sondern auch sozialen Gesichtspunkten. Das Paradigma hinter *Free/Libre and Open Source Software* (FLOSS) basiert auf den Prinzipien Offenheit, Zusammenarbeit und Gemeingutbildung. FLOSS versteht Software als ein öffentliches Gut, das allen zugänglich sein sollte, um Innovationen zu fördern und technologische Abhängigkeiten zu verringern. Die *Open Source Initiative* (2024) definiert FLOSS als Software, deren Quellcode einsehbar, veränderbar und weiter verteilbar ist. Es gibt gute Belege dafür, dass FLOSS einen wichtigen Beitrag zu ökologischerem Technologieeinsatz leisten kann (Viduka 2015). Exemplarisch hervorzuheben ist hier das Betriebssystem GNU/Linux, wodurch z.B. die Lebensdauer von Hardware erheblich verlängert werden kann. Unter sozialen und wirtschaftlichen Gesichtspunkten trägt FLOSS u.a. dazu bei, technologische Abhängigkeiten zu reduzieren und die digitale Souveränität zu stärken.

Refurbished Hardware

Ein großer Hebel für die nachhaltige Konfiguration einer IT-Infrastruktur ist die richtige Auswahl der Hardware. Für den vorliegenden Anwendungszweck (HS-Informatik-Infrastruktur) argumentieren wir für gebrauchte bzw. refurbished Hardware, sowohl für End- als auch Server-Geräte.
Der Lebenszyklus von Hardware lässt sich in drei Phasen einteilen: Herstellung, Betrieb und Entsorgung. Insbesondere bei Endgeräten fällt der größte Teil der CO2e-Emissionen in der Herstellungsphase an (vgl. Prakash et. al. 2016: 21, 26). Allein die mikroelektronischen Komponenten eines Desktop-PCs machen bei einem Nutzungszeitraum von drei Jahren einen Anteil von über 60% der gesamten CO2e aus. Die Verlängerung der Lebenszyklen

eines Geräts reduziert somit nicht nur die Investitionskosten, sondern insbesondere auch die anfallenden CO_2e-Emissionen. Zudem ist der Bedarf an seltenen Erden nicht zu vernachlässigen.
Für Server muss diese Bilanz differenzierter betrachtet werden, da hier weitere Aspekte wie Sekundärnutzung der Abwärme, Auslastung, Anwendungszweck etc. berücksichtigt werden müssen. Allerdings zeigen Bashroush et. al. (2020), dass der Einsatz von refurbished Servern sich nicht nur aus ökologischer, sondern auch aus wirtschaftlicher Sicht lohnt. Das zugrunde liegende Kalkül ist hier, dass das historische Wachstum der Energieeffizienz durch *Moore's Law* in den letzten Jahren deutlich verlangsamt wurde und die meisten Server mit einer Auslastung von ca. 20% laufen. Die Autoren analysieren verschiedene Szenarien zur Erneuerung von Servern und zeigen, dass der Austausch von Servern, die älter sind als 5 Jahre, durch refurbished Geräte wirtschaftlich und ökologisch sinnvoll ist. Für jüngere Server empfiehlt sich eine gezielte Optimierung, wie z.B. durch Speicheraufrüstung.

Green Coding

Green Coding bezeichnet eine Herangehensweise in der Softwareentwicklung, die darauf abzielt, den Energiebedarf und die Umweltbelastung von Software zu minimieren. Dies beinhaltet die Optimierung von Code, Algorithmen und Softwarearchitekturen, um den Ressourcenverbrauch, wie Prozessorleistung, Speicherplatz und Netzwerkbandbreite, zu reduzieren. Darüber hinaus beinhaltet Green Coding eine bewusste Praxis und Haltung in Bezug auf den alltäglichen Entwicklungsprozess in der IT.

Ausgehend von den Diskussionen um die FIfF-Kon 2024 und auf Grundlage des Booklets *Green Coding*, herausgegeben vom Bundesministerium für Umwelt, Naturschutz, nukleare Sicherheit und Verbraucherschutz (2024) haben wir das vorliegende Projekt auf Green-Coding-Aspekte hin überprüft.

In der Green-Coding-Praxis unterscheiden wir zwei Kategorien von Maßnahmen: Erstens solche, die den Entwicklungsprozess und das Umfeld betreffen, und zweitens jene, die darauf abzielen, qualitativ hochwertige und leistungseffiziente Software zu entwickeln.

Zu den *prozessorientierten Ansätze* zählen wir u.a..:

- **Monitoring und Verbrauchsanalyse:** Ressourcenverbrauch (CPU, RAM, Energie) wird erfasst und visualisiert, um Optimierungspotenziale zu erkennen.

- **Dokumentation für bessere Wartbarkeit:** Klare und umfassende Dokumentation von Architektur, Abhängigkeiten und Prozessen sichert die langfristige Wartbarkeit.
- **Schulung der Mitwirkenden:** Weiterbildung durch Fachliteratur, Onlinekurse und interne Schulungen.
- **Modular entwickeln, modular testen:** Eine modulare Softwarearchitektur ermöglicht effizientere Wartung und gezielte Optimierungen.
- **Lastspitzen auf „grüne“ Tageszeiten legen:** Ausführung energieintensiver Prozesse in Zeiten hoher Verfügbarkeit erneuerbarer Energien.

Zu *produktorientierten Ansätzen* zählen u.a.:

- **Effiziente Programmierung:** Der Einsatz optimierter Algorithmen reduziert den Ressourcenverbrauch.
- **Echtzeitverarbeitung von Daten und dynamische Inhalte:** Es wird hinterfragt, ob Echtzeitverarbeitung notwendig ist, das Verarbeiten in größeren Intervallen und in Buffer kann viel Energie sparen, dynamische Inhalte sollten sparsam verwendet werden.
- **Modulare Entwicklung und Code-Reviews:** Softwarearchitektur wird modular aufgebaut, und regelmäßige Code-Überprüfungen sichern Effizienz und Qualität.
- **Datenbank-Indizes und Explain-Plans:** Die Optimierung von Datenbankabfragen reduziert die Rechenlast und verbessert die Performance.
- **Containerisierung:** Anwendungen laufen in separaten Containern, was Skalierbarkeit und Portabilität verbessert.

Wir konnten einige Aspekte identifizieren, die von Beginn an Teil unserer Arbeitspraxis waren. Unter diesen sind beispielsweise zu nennen: Schulung der Mitwirkenden, Containerisierung der Projektinfrastruktur oder Modularität und lose Kopplung der Komponenten. Letzteres ist konzeptionell angelehnt an das Prinzip der Orthogonalität in der Unix-Philosophie (vgl. Raymond 2004: 89).

[Abbildung 8; Bildbeschreibung: *Green Coding Aspekte in unserer Architektur*]

Weiterhin haben wir Aspekte identifiziert, die wir bewusst in das Projekt integrieren (vgl. Abb. 8). Beispiele sind: Als besonders sparsame Methode, um die automatisierten Tests zu beobachten, hat sich früh das Protokoll VNC etabliert. Über VNC wird der entfernte X-Server (auf einem Automation-Container) mit einem Client auf dem Endgerät verbunden. Das Protokoll ermöglicht durch einen Bildbuffer eine visuelle Übertragung, ohne einen kontinuierlichen Videostream zu übertragen.
Die Aspekte Datenerhebung, -Verarbeitung und -Speicherung erfordern zum Einen, sorgfältige und vorausschauende Planung, zum Anderen auch eine fortwährende Überprüfung: Wir haben uns dafür entschieden, die Messdaten für den Zeitraum des Projekts auf einem Datenbank-Server abzulegen, um strukturiert auf die gesamten Testdaten zugreifen zu können. Insbesondere bei der Erhebung und Speicherung der Daten sehen wir Optimierungsbedarf: Anstatt die sekündlich erfassten Daten direkt an die Datenbank zu übertragen, wäre es effizienter, die Datenbank-Verbindung über einen längeren Zeitraum offen zu halten, die Daten z.B. in einem Buffer zwischenzuspeichern und minütlich an den Server zu übertragen.

Weiterhin sei hier auch auf den Open-Source-Gedanken verwiesen, da Quelloffenheit eine Voraussetzung für Anpassbarkeit und Überprüfbarkeit des Codes ist. Um valide Aussagen über die Qualität der eigenen Software treffen zu können, müssen zudem insbesondere externe Abhängigkeiten und Importe wohl bekannt, dokumentiert und gut verwaltet sein.

Viele Aspekte, die hier unter dem Schlagwort Green Coding genannt werden, sind gängige Methoden der Qualitätssicherung. Dies kann und sollte ein Anknüpfungspunkt in der Softwareentwicklung sein. Insbesondere sollte Green Coding als Haltung und Alltagspraxis verstanden werden, in der Handlungen und etablierte Methoden regelmäßig hinterfragt und Entscheidungen anhand fundierter Kenntnisse abgewogen werden können.

Diskussion

Im vorliegenden Projekt haben wir zwei Hauptthemen untersucht: 1. Wie kann allgemein eine nachhaltige IT-Infrastruktur gestaltet werden und 2. Welche Auswirkungen hat im Besonderen ein Remote-Workspace auf den Ressourcenbedarf? Für letztere Fragestellung haben wir uns u.a. mit VSCodium auseinandergesetzt, da dies eine Konfiguration ist, die zunehmend in Anwendungen wie GitHub-Codespaces u.ä. verwendet wird.

Wir haben versucht, uns dem Konzept Nachhaltigkeit ganzheitlich zu nähern und dafür verschiedene Ebenen untersucht (vgl. Abb. 1).

Auf der Technologie-Ebene gibt es gute Indizien für das Nachhaltigkeitspotenzial von gebrauchten Endgeräten, refurbished Servern und ressourcensparender Programmierung zur Verlängerung der Lebensdauer von Hardware. Unberücksichtigt im Betrieb bleibt bisher die Kühlung von Server-Geräten, was insbesondere in unserer Hochschule ein großer Hebel sein könnte.

Auf der Mikroebene liefern die durchgeführten Experimente mit Vim und VSCodium Hinweise auf die Energieeffizienz serverbasierter Entwicklungsumgebungen, was nicht nur Einfluss auf die Hardware-Anforderungen hat sondern auch die Leistungsaufnahme, z.B. im Akkubetrieb, reduziert − ein wichtiges Argument im Hochschulkontext. Hier ist, wie diskutiert, weiterer Forschungsbedarf gegeben, um den Einfluss von Netzwerkverbindungen zu untersuchen. Eine große Herausforderung für langlebige Hardware ist häufig das Thema rund um obsoleszente Soft- und Hardware, dieses kann durch Open Source adressiert werden.

Für die praktische Anwendung bieten die Erkenntnisse aus dem Themenbereich Green Coding wertvolle Einsichten. Zwar fehlt bisher eine systematische Analyse möglicher Reboundeffekte, wir zeigen aber, dass Green Coding mehr ist als eine Anleitung zum Schreiben guter Programme (Mikro-Ebene), sondern darüber hinaus in seiner Praxis − ebenso wie der durch Offenheit und Kollaboration geprägte Open-Source-Gedanke − ein Ansatz sein kann, Verhaltensänderungen zu gestalten und Reboundeffekte zu adressieren (Makro-Ebene). Daher plädieren wir dafür, dass Green-Coding-Herangehensweisen und Open Source auch im Curriculum von Hochschulen fest verankert werden.

Literatur

Bashroush, R., Rteil, N., Kenny, R. and Wynne, A. (2022) 'Optimizing Server Refresh Cycles: The Case for Circular Economy With an Aging Moore’s Law', IEEE Transactions on Sustainable Computing, 7(1), pp. 189-200, 1 Jan.-March. doi: 10.1109/TSUSC.2020.3035234.

Bundesministerium für Umwelt, Naturschutz und nukleare Sicherheit (2024) Green Coding - Booklet zur Workshopreihe. Available at: https://www.bmuv.de/fileadmin/Daten_BMU/Download_PDF/Digitalisierung/das_green_coding_booklet_bf.pdf (Accessed: 23 January 2025).

Köppl-Turyna, M., Briglauer, W., Koch, P., Wolf, M., Schwarzbauer, W., Gotsch, M. and Eberling, E. (2021) 'Digitalisierung und Emissionen'. ECO Austria - Institut für

Wirtschaftsforschung. Available at: https://ecoaustria.ac.at/wp-content/uploads/2021/05/Studie-Digitalisierung-Emissionen.pdf (Accessed: 24 January 2025).

Open Source Initiative (2025) 'Open Source Definition'. Available at: https://opensource.org/osd (Accessed: 23 January 2025).

Prakash, S., Antony, F., Köhler, A. and Liu, R. (2016) Ökologische und ökonomische Aspekte beim Vergleich von Arbeitsplatzcomputern für den Einsatz in Behörden unter Einbeziehung des Nutzerverhaltens. Umweltbundesamt. Available at: https://www.umweltbundesamt.de/sites/default/files/medien/377/publikationen/endbericht_oko-apc_2016_09_27.pdf (Accessed: 1. February 2025).

Raymond, E.S. (2004) The Art of Unix Programming. 1st edn. Addison-Wesley Professional. ISBN: 0131429019.

Viduka, D. and Baić, A. (2015) 'Impact of Open Source software on the environmental protection', Computational Ecology and Software, 5, pp. 113–118.