An: Umweltbundesamt

Wörlitzer Platz 1
06844 Dessau-Roßlau
Deutschland

06.07.2021 - Hamburg, Germany

Angebot für das Projekt „Energieeffizienz-Kennwerte von Komponenten und Werkzeugen der Softwareentwicklung und Vorarbeiten zur Etablierung einer Kennzeichnung für energieeffiziente Software“
FKZ: 37EV 20 102 0, Aktenzeichen 73 012 / 12

Sehr geehrte Damen und Herren,

Bitte finden Sie im folgenden unser Angebot für das ausgeschriebene Projekt. Sowohl das Konzept als auch das Implementierungskonzept wurde gemeinsam mit dem Öko Institut e.V. erarbeitet.

Wir freuen uns auf eine mögliche Zusammenarbeit.

Mit freundlichen Grüßen,

Mit freundlichem Gruß
Max Schulze, Vorstandsvorsitzender (Executive Chairman)

Sustainable Digital
Infrastructure Alliance (SDIA) e.V.
Vereinsregister Hamburg: 24110

Colonnaden 5
20354 Hamburg
Deutschland

contact@sdialliance.org
+49 40 22862448

Zusammenfassung

Digitale Technologien, und damit zum Großteil Software werden von vielen Nationen gefördert, um den Klimawandel zu stoppen, sowie den globalen Handel und Wertschöpfungsketten zu verbessern und transparenter zu machen. Dabei ist der reale Ressourcenverbrauch von Software kaum messbar - wir sehen jedoch im massiven Wachstum der physischen Infrastruktur bereits, dass der reale Ressourcenverbrauch wahrscheinlich wesentlich höher ist als angenommen.

Methoden für die Messung von Ressourcenverbräuchen wurden bereits durch das Umweltbundesamt[1], das Öko-Institut[2] und den Umweltcampus Birkenfeld[3] entwickelt und können im Rahmen dieses Projektes angewandt werden. Vielmehr liegt der Fokus auf der Veränderung des Bewusstseins von Softwareentwickler*innen. Denn die Software-Gemeinschaft nimmt heutzutage grundsätzlich an, dass Software keinen wirklich großen Energieverbrauch hat, beziehungsweise dieser so klein ist, dass er nicht in Entscheidungen bei der Entwicklung einfließen muss.

Daher geht es in unserem Angebot primär um die Einbettung einer Kennzeichnung in die Arbeitsumgebungen von Softwareentwickler*innen, um eine bessere Sichtbarkeit für die Energieeffizienz und das Schaffen einer einfachen Möglichkeit, durch die Einbettung einer Kennzeichnung in gängige Software-Bibliotheken bessere Entscheidungen bei der Architektur von Software zu treffen.

Als Fallbeispiel, dass Energieeffizienz in Software sinnvoll und machbar ist, kann die Entwicklung von Anwendungen für Smartphones herangezogen werden. Hier wird bereits bei der Entwicklung stark darauf geachtet, den Energieverbrauch zu minimieren, da ein hoher Energieverbrauch zu einer beeinträchtigten Erfahrung für den Nutzer führt und die Zeitspanne zwischen Ladezyklen der Batterie verkürzt. Daher werden z.B. in den Entwicklungswerkzeugen für das Android Betriebssystem oder Apple iOS sowohl Energieeffizienz-Metriken angezeigt als auch Hilfestellungen und Tipps gegeben, welche es ermöglichen, die Energieeffizienz der Software zu steigern. Dies beweist, dass durch das Hinzufügen einer Einschränkung (Batterie) und das Bereitstellen von Hilfestellungen und Kennzahlen das Verhalten von Softwareentwickler*innen und damit auch die Architektur der Software verändert werden kann. Dieses Modell dient uns als Referenz für das Vorgehen in diesem Projekt.

---

[1] UBA 2018: Entwicklung und Anwendung von Bewertungsgrundlagen für ressourceneffiziente Software unter Berücksichtigung bestehender Methodik: www.umweltbundesamt.de/presse/pressemitteilungen/umweltvertraeglichkeit-von-software-ist-jetzt

[2] Jens Gröger, Bits & Bäume 2018: Einfluss von Software auf den Ressourcenverbrauch: https://www.oeko.de/fileadmin/oekodoc/Einfluss-von-Software-auf-Ressourcenverbrauch.pdf

[3] Umweltcampus Birkenfeld (2018). Refoplan 2018: https://www.umwelt-campus.de/forschung/projekte/green-software-engineering/projekte/refoplan-2018

Um den Erfolg dieses Vorhabens zu erhöhen, teilen wir unsere Arbeit in drei Fokusbereiche aus Sicht der Software-Entwicklung ein. Dafür ist es wichtig hervorzuheben, dass Softwareentwicklung sich in zwei Teilbereiche gliedert:

- **Komponentenentwicklung:** Die Erstellung von wiederverwendbaren Software-Komponenten (wie z.B. für eine Anmeldung, die Kommunikation mit einer Datenbank, das Anzeigen eines Formulars); oft entwickelt im Rahmen von Open Source Projekten und daher gepflegt von freiwilligen Entwickler*innen.

- **Anwendungsentwicklung:** Die Erstellung einer Software-Anwendung für einen spezifischen Anwendungsfall, wie z.B. eine Warenhausverwaltung oder einen Online-Shop; hier werden zum Großteil bestechende wiederverwendbare Komponenten (s.o.) integriert und miteinander verbunden. Die Einbettung von sehr vielen Komponenten und Bibliotheken, von denen aber nur vereinzelte Funktionen genutzt werden, sorgt für die Entstehung des sog. Software-Bloatings.

Auf Basis dieser Teilbereiche, teilt sich unser Fokus entsprechend auf:

- Die **sichtbare und aktuelle Kennzeichnung** hinsichtlich Energieeffizienz von Software Komponenten, sodass Softwareentwickler*innen bei der Auswahl der Komponenten die Möglichkeit haben, die Effizienz in ihre Entscheidung mit einfließen zu lassen

- Die **Bereitstellung von Kennzahlen und Hilfestellungen in der Entwicklungsumgebung** für die Anwendungsentwicklung, z.B. durch die Integration in gängige Entwicklungsumgebungen wie Eclipse, IntelliJ, Atom, u.a. oder in sog. Kontinuierlichen Integrationssystemen, bei denen die Software bei jeder Veränderung auf Mängel geprüft wird.

- Die **Kennzeichnung von Software-Anwendungen** hinsichtlich ihrer Energieeffizienz bei der Herausgabe und Veröffentlichung in der Form eines nationalen Bewertungssystems.

Um eine möglichst Weite Verbreitung zu gewährleisten, werden wir in unserer Arbeit auf einer gängigen Vorgehensweise in der Softwareentwicklung aufbauen und die Messung der Energieeffizienz in bestehende „Linter“[4] einbauen. Es ist in fast jedem Software Projekt - sowohl in der Komponenten als auch in der Anwendungsentwicklung - üblich, die Qualität des Programmcodes kontinuierlich zu überwachen und zu messen. Dafür wurden sog. „Linter“ entwickelt, die während der Entwicklung, oder

[4] Macdonald, F., Miller, J., Brooks, A., Roper, M. and Wood, M., 1995, July. A review of tool support for software inspection. In Proceedings Seventh International Workshop on Computer-Aided Software Engineering (pp. 340-349). IEEE.

spätestens als der automatisierten Qualitätsüberwachung, den Programmcode überprüfen und mögliche Schwachstellen oder Anzeichen von schlechter Qualität identifizieren[5].

Diese „Linter“ sind in fast allen Programmiersprachen verfügbar und werden bereits von einem Großteil der Softwareentwickler*innen-Gemeinschaft verwendet[6]. „Linter“ sind außerdem bereits in fast allen Entwicklungsumgebungen eingebaut, sodass Entwickler*innen mit einem Knopfdruck die Qualität (und durch unsere Erweiterung auch der Energieeffizienz) der Anwendung prüfen können[7]. Solche „Linter“ um die Fähigkeit zu erweitern, auch die Energieeffizienz anzuzeigen und als quelloffene Erweiterung (open source) durch das Umweltbundesamt bereitzustellen, führt unseres Erachtens nach zu der höchstmöglichen Verbreitung und liefert damit einen großen Schritt hin zu einer weiten Wahrnehmung und Sensibilisierung für die Effizienz von Software.

Unser Ansatz, die bestehenden Methoden vom Umweltbundesamt, dem Öko-Institut e.V. sowie dem Umweltcampus Birkenfeld, die bereits erwähnt wurden, in eine quelloffene Erweiterung von „Linter“-Komponenten und Werkzeugen zu implementieren, führt zu einer gleichzeitigen Lösung von sowohl der Kennzeichnung von Komponenten als auch zur Kennzeichnung und Hilfestellung in der Anwendungsentwicklung, denn die bestehenden Linter-Werkzeuge kommen in beiden Bereich gleichermaßen zum Einsatz. Zusätzlich kann mit diesem Ansatz eine Bewertung als Teil des Software-Auslieferungsprozesses (Continuous Integration[8]) implementiert und die Effizienz der Software durch eine grafische Kennzeichnung dargestellt werden.

Daher ist unser Vorschlag ebenso in sich selbst effizient, da wir anstatt neue Software und Komponenten zu bauen, die Integration mit bestehenden Werkzeugen bevorzugen und damit sowohl die Nutzbarkeit und Verbreitung erhöhen als auch vermeiden, dass ein weiteres Werkzeug gebaut und verwendet werden muss.

Um ein nationales Bewertungssystem zu realisieren, wäre es außerdem aus unserer Sicht sinnvoll, Software-Entwickler*innen die Möglichkeit zu geben, als Teil der Messung bei der finalen Auslieferung der

---

[5] J. Gimpel, "Software That Checks Software: The Impact of PC-lint," in IEEE Software, vol. 31, no. 1, pp. 15-19, Jan.-Feb. 2014, doi: 10.1109/MS.2014.13.
[6] Gwodz, M., 2020: Linters aren’t in your way. They’re on your side: https://stackoverflow.blog/2020/07/20/linters-arent-in-your-way-theyre-on-your-side/
[7] Tómasdóttir, K.F., Aniche, M. and van Deursen, A., 2017, October. Why and how JavaScript developers use linters. In 2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE) (pp. 578-589). IEEE.
[8] Sacolick, I., 2020: What is CI/CD? Continuous integration and continuous delivery explained: https://www.infoworld.com/article/3271126/what-is-cicd-continuous-integration-and-continuous-delivery-explained.html

Software (Continuous Delivery[9]) die Messdaten an eine unabhängige Stelle zu leiten und dort die Messung zu verifizieren und zu veröffentlichen. Dies ist insbesondere bei der Entwicklung von Software-Komponenten in Bezug auf die Qualität des Programmcodes bereits üblich, und wird von den meisten quelloffenen Komponenten als Qualitätssiegel verwendet[10]. Die Machbarkeit eines solchen Systems, werden wir gemeinsam mit dem Umweltbundesamt im Rahmen dieses Projektes verifizieren.

Aus Sicht unserer Nachhaltigkeits-Allianz ist die Kennzeichnung von Software hinsichtlich Energie- und Ressourcenverbrauch ein elementarer Bestandteil um die Umweltverträglichkeit der zugrundeliegenden physischen Infrastruktur (wie z.B. Rechenzentren und ICT Hardware) zu verbessern. Die Reduktion von Energieverbrauch und Umweltauswirkung der gesamten Wertschöpfungskette von Software ist ein kritischer Bestandteil für die Erreichung unserer gemeinsamen Klimaziele, insbesondere wenn für ihre Erreichung Software einsetzen wollen (siehe Scope 2 Emissionen).

[9] Sacolick, I., 2020: What is CI/CD? Continuous integration and continuous delivery explained: https://www.infoworld.com/article/3271126/what-is-cicd-continuous-integration-and-continuous-delivery-explained.html

[10] Humble, J. and Molesky, J., 2011. Why enterprises must adopt devops to enable continuous delivery. Cutter IT Journal, 24(8), p.6.

AP1: Messung der Energieeffizienz und Hardware-Inanspruchnahmen von Komponenten und Werkzeugen der Softwareentwicklung

Das erste Arbeitspaket wird in Zusammenarbeit mit dem Öko-Institut bearbeitet. Das Öko-Institut bringt die Erfahrungen und Ergebnisse aus der vorherigen Arbeit des Umweltbundesamtes[11] in das Projekt ein und bildet die Grundlage der Messmethodik. Außerdem werden die bestehenden quelloffenen Komponenten OSCAR[12] und ESSD[13] vom Umwelt Campus Birkenfeld mit einbezogen und liefen einen Teil der zugrundeliegenden Technologie für die Implementierung der Messung für Anwendungssoftware, Bibliotheken und Komponenten.

Durch die Zusammenarbeit mit dem Öko-Institut ist zudem sichergestellt, dass das Projekt die bestehende Arbeit effizient einbetten kann und der Fokus auf die Implementierung und Verbreitung gelegt werden kann.

Erarbeitung von vereinfachten Leitfäden für Programmiersprachen

Um es Software-Entwickler*innen möglichst einfach zu machen, den Kriterienkatalog für nachhaltige Software in eigenen Projekten anzuwenden, erarbeiten wir in einem ersten Schritt vereinfachte Leitfäden, die die Ergebnisse des aus dem UFOPLAN-Projekt „Sustainable Software Design – Entwicklung und Anwendung von Bewertungsgrundlagen für ressourceneffiziente Softwareprodukte unter Berücksichtigung bestehender Methodik"[14] und die Mindestanforderungen für umweltverträgliche Software aus dem blauen Engel[15] einbeziehen und Handlungsempfehlungen für die ausgewählten Programmiersprachen ableiten. Diese Leitfäden umfassen zwischen 5-10 Seiten, die im Rahmen dieses Angebots enthalten sind, für jede der 6 Programmiersprachen und wird als README (Textdatei) und PDF bereitgestellt.

Gleichzeitig nutzen wir diesen Arbeitsschritt, um die Lücke zwischen der Bewertungsmethodik und den Anforderungen an eine umweltverträgliche Software zu schließen und die Informationslage über die

[11] UBA (2018): Umweltverträglichkeit von Software ist jetzt messbar: https://www.umweltbundesamt.de/presse/pressemitteilungen/umweltvertraeglichkeit-von-software-ist-jetzt
[12] https://oscar.umwelt-campus.de/
[13] https://gitlab.umwelt-campus.de/a.guldner/essd
[14] Umweltcampus Birkenfeld: Kriterienkatalog für nachhaltige Software: https://www.umwelt-campus.de/forschung/projekte/green-software-engineering/kriterienkatalog/einleitung
[15] Blauer Engel (2020): Ressourcen- und energieeffiziente Softwareprodukte

Energieeffizienz von Komponenten und Werkzeugen der Softwareentwicklung unter Software-Entwickler*innen zu verbessern.

Auswahl von Programmiersprachen und Entwicklerumgebungen (AP 1.1)

Wie in der Ausschreibung gewünscht, werden wir im folgenden zwei weitere Programmiersprachen und zusätzliche Entwicklungsumgebungen in das Projekt einfließen lassen.

Für die Programmiersprachen, werden neben den in der Ausschreibung geforderten Sprachen auch **Go und JavaScript** mit in das Projekt einbezogen.

- **Go Lang**[16]: Eine mittlerweile sehr anerkannte Sprache die besonders bei der Entwicklung von operativen Umgebungen für Container-basierte Anwendungen eingesetzt wird. Die Sprache ist sehr bekannt, da sie für die Entwicklung von Kubernetes (ebd.), einer Software die für die Orchestrierung der meisten Cloud-Umgebungen verwendet wird. Go wird in dieses Angebot einbezogen,, um gerade Cloud-Umgebungen die Möglichkeit zu geben, direkt in der Entwicklung der serverseitigen Komponenten, Energieeffizienz zu berücksichtigen und umweltverträgliches Software Design zu fördern.

- **JavaScript**[17]: Die populärste Sprache für die Entwicklung von web-basierten Anwendungen, die vor allem in einer Internet-Browser Umgebung ausgeführt werden. Mittlerweile findet sich JavaScript aber auch auf der Serverseite wieder, in Form von Node.JS. JavaScript wird mittlerweile vielseitig eingesetzt und ist besonders anfällig für Software-Bloating, da es hier besonders viele und oft duplizierte Bibliotheken und Komponenten gibt.

Im Bereich der Software Umgebungen wird außerdem die sehr bekannte kommerzielle Entwicklungsumgebung aus dem Hause **Jetbrains**[18] **und Visual Studio Code**[19] **bzw. GitHub Atom**[20] neben den geforderten Umgebungen aus der Ausschreibung einbezogen. Beide sind weit verbreitet und haben einige interessante Besonderheiten:

- Jetbrains ist mit IntelliJ sowohl für Java, als auch C/C++ und Python eine der am weitesten verbreiteten kommerziellen Entwicklungsumgebungen. Interessant im Bezug auf unser Vorhaben ist besonders die Kommunikation des Unternehmens hinsichtlich Ressourcen- und

[16] GO (2019): Go for Cloud & Network Services: https://go.dev/solutions/cloud/
[17] Mozilla (2021): Java Script: https://developer.mozilla.org/en-US/docs/Web/JavaScript
[18] https://www.jetbrains.com/
[19] https://code.visualstudio.com/
[20] https://atom.io/

Energieeffizienz und der Fokus auf Geschwindigkeit in der Entwicklung (siehe Website in der Fußnote). Mit Hilfe der Messverfahren aus Arbeitspaket 1 lassen sich die Aussagen der Hersteller validieren und möglicherweise Handlungsempfehlungen bei der Verwendung für Software-Entwickler*innen ableiten.

- Visual Studio Code und GitHub Atom, basieren auf derselben Electron-basierten[21] technischen Plattform. Es wird für die Darstellung der Oberfläche eine Version des Google Chrome Webbrowsers verwendet und im Hintergrund geladen. Die gesamte Entwicklungsumgebung ist in JavaScript entwickelt und passt daher gut zur Auswahl der Programmiersprachen im vorherigen Absatz.

Weder für die geforderten Entwicklungsumgebungen noch für die von uns zusätzlich betrachteten Umgebungen fallen Lizenzgebühren an, da diese entweder auf Open Source basieren oder für gemeinnützige Organisationen kostenfrei zur Verfügung stehen.

Architektur und Aufbau eines skalierbaren Messlabors für Software, Bibliotheken, Komponenten und Entwicklungsumgebungen (AP 1.2)

Ziel muss es sein, ein skalierbares, zuverlässiges und reproduzierbares Messlabor zu implementieren, welches quelloffen auch von Software-Entwickler*innen selbst verwendet werden kann, um das Messverfahren des Umweltbundesamtes in eigenen Projekten anzuwenden.

Daher werden wir, das Testlabor mit Hilfe von bestehender, gängiger quelloffener Software aus dem Bereich der kontinuierlichen Integration[22] wie z.B. Jenkins umsetzen. Als Teil des Projektes entwickeln wir eine Komponente für eine bestehende Umgebung, in diesem Falle Jenkins, mit der die Teststrecke auf Basis von verschiedenen Quellcodes getestet werden kann. Damit ist es möglich, verschiedene Quellcode-Speicher (Repositories) zu hinterlegen und das Messverfahren automatisiert anzuwenden.

Da für die Laborumgebung unsere bereits bestehende Infrastruktur verwendet wird und quelloffene Server- Hardware genutzt wird, fallen keine zusätzlichen Kosten für den Erwerb von Sachmitteln an. Die Nutzung der bestehenden Messlabore der SDIA sind im vollem Umfang in diesem Angebot enthalten.

[21] https://brainhub.eu/library/what-is-electron-js/
[22] Sacolick, I., 2020: What is CI/CD? Continuous integration and continuous delivery explained: https://www.infoworld.com/article/3271126/what-is-cicd-continuous-integration-and-continuous-delivery-explained.html

Um das Messlabor in sich selbst wiederholbar und übertragbar zu machen, werden wir die Anwendungen und Komponenten mit Hilfe von Software-Containern bereitstellen und eine Kubernetes Umgebung einbetten. So ist es möglich, die gesamte Testumgebung und das Labor mit wenigen Schritten zu starten und im Selbsttest die Funktionsfähigkeit und Integrität zu prüfen. Dieses Vorgehen ermöglicht es außerdem nach Abschluss des Projektes, die Ergebnisse, samt der Messlabore und Teststrecken, auf andere Betriebsumgebungen zu übertragen, z.B. beim Umweltbundesamt. Die entsprechende Dokumentation bzw. Handbuch, werden als Teil des Angebots mitgeliefert.

Außerdem werden sowohl das Messlabor als auch die entwickelten Module für Jenkins oder andere Werkzeuge aus dem Bereich der kontinuierlichen Integration nach Absprache mit dem UBA veröffentlicht, sodass Software-Entwickler*innen die Möglichkeit haben, die Werkzeuge in ihre eigenen Prozesse zu integrieren.

Implementierung Mess- und Analyse-Software (AP 1.2)

Das Messlabor, welches im vorherigen Abschnitt erläutert wurde, verwendet als Grundlage die Mess- und Analysesoftware, die als Teil des Arbeitspakets 1 entwickelt wird. Um eine hohe Kompatibilität zu gewährleisten, wird diese Software auf bestehenden Werkzeugen der Quellcode-Qualitätsmessung (Linter-Werkzeuge) aufgesetzt. Diese erheben bereits viele der benötigten Eingangsdaten die für die Anwendung des bereits entwickelten Messverfahrens notwendig sind. Zusätzlich verfügen diese Linter-Werkzeuge über eine sehr hohe Verbreitung - es bestehen bereits sowohl Integrationen für alle gängigen Entwicklungsumgebungen und kontinuierliche Integrationsplattformen, als auch für die manuelle Verwendung von Software-Entwickler*innen als Teil des Entwicklungsprozesses.

Die Anforderungen an die Mess- und Analysesoftware, insbesondere

- die Unterstützung der Messung: Einlesen der Messdaten und -protokolle,
- das Absichern der Eingabedaten, so dass unzulässige oder unplausible Messdaten, automatisch markiert oder blockiert werden,
- die schrittweise, verständliche Führung durch die energetische Bilanzierung,
- die Analyse und Präsentation der importierten Messdaten in einer grafischen Oberfläche
- die Auswertung variiert die Anzahl der auszuwertenden Datensätze und der darzustellenden Ergebnisse
- die graphische Ausgabe der Ergebnisse - mit kurzer Erläuterung,
- Datenverwaltungsfunktionen, wie Speichern, Exportieren von Werten und Ergebnissen,
- integrierte Bedienhinweise

können mit dem Vorgehen kosteneffizient und mit hoher Kompatibilität umgesetzt werden, und bilden damit eine sinnvolle Basis für eine weitere Verbreitung der Ergebnisse aus diesem Projekt. Durch die Verwendung von bereits üblichen Programmen reduziert sich der Anpassungsaufwand für Software-Entwickler*innen und die Wahrscheinlichkeit für eine erfolgreiche, breitflächige Nutzung in der Gemeinschaft wird drastisch erhöht.

Durchführung der Messungen (AP 1.3)

Auf Basis der Herangehensweise aus dem Aufbau des Messlabors ist es möglich, die Messungen aus dem Arbeitspaket 1.3 zuverlässig und systematisch durchzuführen. Da die Messdurchläufe automatisiert sind, können diese beliebig wiederholt und verschiedene Beispieldatensätze effizient miteinander verglichen werden.

Für die Durchführung der Tests - falls vom Umweltbundesamt im Projektverlauf nicht anders definiert - werden jeweils die populärsten (gemessen an den Aktivitäten auf Github) Bibliotheken und Software-Komponenten für jede Programmiersprache untersucht. Dafür wird für jede Komponente eine Skelett-Software erstellt (“Testobjekt”), die dann im Messaufbau für die Tests verwendet wird. Da jede Programmiersprache über eigene Konfigurationen verfügt, die die Energieeffizienz beeinflussen können, wird jedes Testobjekt in verschiedenen Konfigurationsvarianten (“Varianten”) im Messaufbau geprüft. Diese Arbeiten sind im Angebot bereits inkludiert. Die jeweiligen Testdurchläufe für jedes Testobjekt können beliebig oft wiederholt werden, da sie automatisiert sind.

Ein Testdurchlauf ist als folgende (vereinfachte) Prozesskette (die in unserem Labor automatisiert ist) definiert:

- Starten der Software
- Messung der Energiekennzahlen über die DCIM Schnittstelle von der Hardware
- Messung der Energiekennzahlen über die Strommesser (externe Validierungsdaten für die DCIM Daten)
- Erfassen von Laufzeitinformationen (Auslastung CPU, Speicher, etc.)
- Stoppen der Software

Konkret bedeutet das für die jeweiligen Programmiersprachen:

- Python: mindestens 5 Testobjekte, die jeweils in 3 Varianten getestet werden; min. 100 Durchläufe je Testobjekt/Varianten Kombination.

- Java: mindestens 5 Testobjekte, die jeweils in 3 Varianten getestet werden; min. 100 Durchläufe je Testobjekt/Varianten Kombination.
- C: mindestens 3 Testobjekte, die jeweils in 2 Varianten getestet werden; min. 100 Durchläufe je Testobjekt/Varianten Kombination.
- C++: mindestens 3 Testobjekte, die jeweils in 2 Varianten getestet werden; min. 100 Durchläufe je Testobjekt/Varianten Kombination.
- Go Lang: mindestens 5 Testobjekte, die jeweils in 2 Varianten getestet werden; min. 100 Durchläufe je Testobjekt/Varianten Kombination.
- JavaScript: mindestens 5 Testobjekte, die jeweils in 2 Varianten getestet werden; min. 100 Durchläufe je Testobjekt/Varianten Kombination.

Insgesamt werden 26 verschiedene Software-Testobjekte erstellt, untersucht und 6.200 Durchläufe im Labor durchgeführt.

Für die 4 Entwicklungsumgebungen, erstellen wir im Rahmen des Angebots 3 Testprojekte (3-stufig mit steigender Komplexität des Programmcodes, der in der Entwicklungsumgebung entwickelt wird). Diese Testprojekte werden für jede Entwicklungsumgebung abgestimmt (z.B. nach Programmiersprache) und auch in diesem Test werden Konfigurationsvarianten der Entwicklungsumgebung untersucht.

- Android-Studio, 3 Testprojekte, die jeweils in 2 Varianten getestet werden; min. 100 Durchläufe je Testobjekt/Varianten Kombination.
- Visual Studio, 2 Testprojekte, die jeweils in 2 Varianten getestet werden; min. 100 Durchläufe je Testobjekt/Varianten Kombination.
- Eclipse, 5 Testprojekte in mehreren Programmiersprachen, die jeweils in 2 Varianten getestet werden; min. 100 Durchläufe je Testobjekt/Varianten Kombination.
- Jetbrains IntelliJ, 2 Testprojekte, die jeweils in 2 Varianten getestet werden; min. 100 Durchläufe je Testobjekt/Varianten Kombination.
- GitHub Atom, 5 Testprojekte in mehreren Programmiersprachen, die jeweils in 2 Varianten getestet werden; min. 100 Durchläufe je Testobjekt/Varianten Kombination.

Insgesamt werden 17 Testprojekte erstellt, untersucht und 3.400 Durchläufe im Labor durchgeführt.

Durch die Verwendung bestehender „Linter“-Werkzeuge und existierende kontinuierliche Integrationssysteme können die bestehenden Visualisierungsfunktionen genutzt werden, um die Ergebnisse der verschiedenen Messdurchläufe grafisch darzustellen und Vergleiche anzustellen.

Zusätzlich können durch die Veröffentlichung des Messaufbaus, und im Einklang mit der Kommunikationsstrategie aus dem Arbeitspaket 3, Gemeinschaften von Software-Entwickler*innen

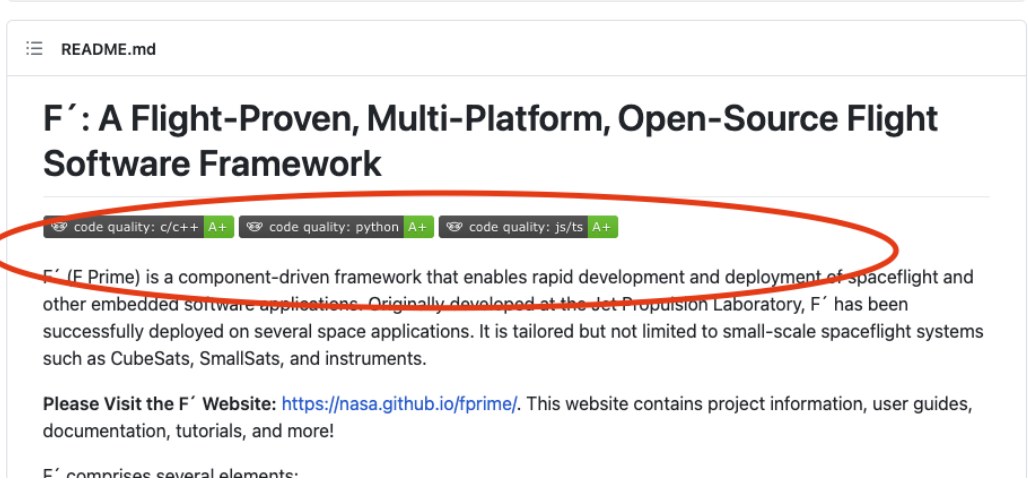
angeregt werden, ihre eigenen Komponenten, z.B. auf GitHub oder GitLab, zu messen und die Messergebnisse dem Projekt zur Verfügung zu stellen. So kann die Zuverlässigkeit der Messung und des Messaufbaus zusätzlich überprüft werden, insbesondere auf die Ausführung in verschiedenen Laufzeitumgebungen oder verschiedener Hardware Konfigurationen.

AP2: Energieeffizienzbewertung von Software

In diesem Arbeitspaket werden die Ergebnisse aus den eingangs erwähnten bestehenden Projekten zur Methodenentwicklung für energieeffiziente Software zusammengeführt und im Einklang mit den Kriterien des Blauen Engels gebracht. Aus der Arbeit ergibt sich im ersten Schritt eine Bewertungsmethode und eine Liste von Handlungsempfehlungen die, je nach Programmiersprache und Messergebnis, den Software-Entwickler*innen eine sinnvolle Unterstützung für die Verbesserung des Quellcodes bietet.

Im zweiten Schritt wird eine Kennzeichnung erarbeitet, die in einem ersten Pilotversuch in öffentlichen Bibliotheken und Komponenten, z.B. auf GitHub, GitLab, Maven, NPM oder anderen öffentlichen Komponenten-Datenbanken eingebunden werden kann und somit die Sensibilisierung für energieeffiziente Software-Entwicklung erhöht.

Ein Beispiel für die bereits in der Open Source Gemeinschaft gängige Kennzeichnung findet sich im folgenden Schaubild:



Quelle: https://github.com/nasa/fprime

Umsetzung der Energieeffizienzbewertung für den Prozess der Softwareentwicklung

Durch den „Linter“-Ansatz bei der Entwicklung des Mess- und Analysetools im Arbeitspaket 1 werden die Anforderungen hinsichtlich der Einbeziehung von anderen Qualitätsstandards, wie der internationalen Normenreihe ISO/IEC 25010, ohne Beschränkungen umgesetzt. Zusätzlich werden durch den Ansatz bereits bestehende Standards und Richtlinien, die nicht genormt sind, jedoch z.B. Programmiersprachen-spezifisch als Standard gelten, in die Berechnung des Energieeffizienzindex einfließen.

Besonders wichtig ist hierbei eine Berechnung, die spezifisch für die jeweiligen Programmiersprachen ist. Insbesondere bei den kompilierten und interpretierten Programmiersprachen können die Energieeffizienz-Messungen in Sachen Laufzeit bzw. bei der Kompilierung stark variieren. Zusätzlich ist ein Energieeffizienzindex nur dann für Software-Entwickler*innen handlungsrelevant, wenn er akkurat, spezifisch und beeinflussbar ist. Durch die Veränderung und Optimierung des Programmcodes bzw. durch die Umsetzung der Handlungsempfehlungen muss eine Verbesserung des Energieeffizienzindex bei der nächsten Messung sofort sichtbar sein.

Ziel des Arbeitsschrittes ist es, entsprechende Programmiersprachen-spezifische Eigenschaften und Kriterien zusammenzutragen und diese bereits an mögliche Handlungsempfehlungen zu koppeln. Zusätzlich wird eine Reihe von Testberechnungen mit Hilfe des Messlabors realisiert, um die richtige Gewichtung der Kriterien und Indikatoren für eine sinnvolle Indizierung zu ermitteln.

Die Ausarbeitung, Untersuchung und Berechnung von Programmiersprachen-spezifischen Energieeffizienzindizes, sowie die Ausarbeitung der entsprechenden Handlungsempfehlungen sind Teil dieses Angebots.

Software-Werkzeug für die Unterstützung von Software-Entwickler*innen

Wie bereits in den vorherigen Arbeitspaketen beschrieben, wird die Umsetzung der Software zur Hilfestellung der Software-Entwickler*innen nicht als eigenständige Software realisiert, sondern auf den Programmiersprachen-spezifischen bestehenden „Linter“-Werkzeugen aufgebaut, die bereits ein fester Bestandteil des Arbeitsablaufes von Software-Entwickler*innen sind und bereits viele der benötigten

Indikatoren und Messwerte zusammenstellen - z.B. die Komplexität des Programmcodes, Wiederholungen und ungenutzte Bibliotheken.

Die Implementierung in die bestehenden „Linter"-Werkzeuge ist außerdem effizienter, da sie auf jahrelange Optimierungen von bestehenden Code-Analyse Algorithmen aufbaut, die in der internationalen Open Source Gemeinschaft entwickelt worden sind.

Beispielhaft haben wir einige „Linter"-Werkzeuge für die Programmiersprachen aus den Projektanforderungen zusammengestellt. Die Auswahl der „Linter"-Werkzeuge für die Implementierung einer Erweiterung, die die Energieeffizienz-Messung des Umweltbundesamtes implementiert, erfolgt in Abstimmung mit den Projektansprechpartnern.

- Python: PyLint[23]
- Java: Lint4J[24]
- C/C++: CPPCheck[25]
- Go Lang: Go Lang CI Lint[26]
- JavaScript: JS Hint[27]

Diese Werkzeuge sind bereits in den angestrebten Entwicklungsumgebungen integriert und werden von vielen Entwicklern verwendet. Einige Beispiele:

- Visual Studio: Visual Studio Lint[28]
- Eclipse: Coala[29]
- Visual Studio Code: Linting for Python[30]

Die Entwicklung der entsprechenden Erweiterungen für die genannten „Linter"-Werkzeuge der ausgewählten Programmiersprachen sind im Angebot enthalten.

Machbarkeit einer Kennzeichnung für energieeffiziente Software

Durch die vorhergegangene Implementierung einer Erweiterung in die bestehenden Prozesse von Software-Entwickler*innen ist eine Kennzeichnung ein folgerichtiger Schritt, insbesondere für

[23] https://pylint.org
[24] http://www.jutils.com
[25] http://cppcheck.sourceforge.net
[26] https://github.com/golangci/golangci-lint
[27] https://github.com/jshint/jshint
[28] https://marketplace.visualstudio.com/items?itemName=Anna-JayneMetcalfe.VisualLint
[29] https://marketplace.eclipse.org/content/coala-eclipse-plug
[30] https://code.visualstudio.com/docs/python/linting

Software-Komponenten und Bibliotheken, die bei der Anwendungsentwicklung eingebunden werden. Hier ermöglicht eine Kennzeichnung den Anwendungsentwickler*innen bei der Auswahl von Komponenten auch die Energieeffizienz mit einzubeziehen. Ohne eine solche Entscheidungshilfe wird eine energieeffiziente Anwendungsentwicklung komplex, und ein Großteil des Programmcodes, der sich nicht in der Anwendung selbst befindet sondern in den Bibliotheken und Komponenten, kann nicht optimiert werden.

Wir werden bei der Prüfung der Machbarkeit einen Pilotversuch unternehmen und das automatisierte Messlabor nutzen, um eine Schnittmenge von öffentlichen Quellen, z.B. auf GitHub oder GitLab hinsichtlich der Energieeffzienz mit Hilfe des entwickelten Messwerkzeugs zu prüfen und eine Kennzeichnung in Form einer „Badge“ (siehe Schaubild) den Entwicklern*innen, die den Programmcode weiterentwickeln, bereitzustellen.

Zusätzlich werden wir bei dem Pilotversuch Partner wie die Eclipse Foundation, die Cloud Native Foundation und die Apache Foundation mit einbeziehen, um deren Open Source Komponenten hinsichtlich Energieeffizienz zu prüfen und in möglicherweise in Folgeprojekten gemeinsam zu verbessern. Diese drei Stiftungen verwalten die am meisten verwendeten öffentlichen Komponenten und Bibliotheken, die an fast allen Software Anwendungen weltweit zum Einsatz kommen.

Grundsätzlich lässt sich das Vorgehen in diesem Arbeitspaket in drei Schritte gliedern:

1. Im ersten Schritt erfassen wir die bestehenden Metriken hinsichtlich der Effizienz aus den verschiedenen Programmiersprachen. Wie im Angebot erklärt, gibt es bereits Ansätze in den Entwickler-Gemeinschaften, die in die Analyse der Machbarkeit mit einfließen werden.

2. Im zweiten Schritt beziehen wir die die Ergebnisse aus den verschiedenen Messungen ein und nutzen die Ergebnisse, um mögliche Handlungsempfehlungen für die verschiedenen Programmiersprachen zu identifizieren - und prüfen dadurch die Machbarkeit einer Kennzeichnung. Dies erfolgt durch die Durchsicht der Messdaten und Visualisierung, um mögliche Muster zu erkennen, sowie technische Experimente an der Software (Testobjekte) selbst um die Muster die verstehen und Empfehlungen ableiten zu können.

3. Im dritten Schritt, um unsere Annahmen zu validieren, verändern wir die Software die im Messaufbau erarbeitet wurde mit den erarbeiteten Handlungsempfehlungen und möglichen Ansätzen, um die Software energieeffizienter zu machen. In unserem automatisierten Messaufbau wiederholen wir einige unserer Tests und prüfen damit, ob und inwiefern die Veränderungen zu einer messbaren Veränderung führen.

Aus diesem Vorgehen erarbeiten wir die konzeptionellen Vorschläge für eine Kennzeichnung und geben klare Implikationen zu weiterem Forschungsbedarf, Handlungsbereichen und -empfehlungen. Teil der konzeptionellen Ausarbeitung sind auch visuelle Vorschläge und Adaptionsstrategien für eine Verbreitung einer möglichen Kennzeichnung.

Sustainable Digital
Infrastructure Alliance (SDIA) e.V.
Vereinsregister Hamburg: 24110

Colonnaden 5
20354 Hamburg
Deutschland

contact@sdialliance.org
+49 40 22862448

AP3: Kommunikationsplan und Verbreitung der Ergebnisse

Um eine hohe Verbreitung der Ergebnisse zu gewährleisten, setzen wir bei der Kommunikation auf eine mehrschichtige Strategie aus eigenen Maßnahmen und die Aktivierung von Partnernetzwerken.

Es ist jedoch besonders hervorzuheben, dass die Motivation der Sustainable Digital Infrastructure Alliance e.V. (SDIA), sich auf das Projekt gemeinsam mit dem Öko Institut e.V. zu bewerben, darin begründet ist, dass wir in unseren eigenen Forschungen immer wieder zu dem Schluss gekommen sind, dass insbesondere die Architektur und Entwicklung von Software ausschlaggebend für die Realisierung einer nachhaltigen Digitalwirtschaft und digitalen Technologien ist. In der heutigen internationalen Gemeinschaft von Software-Entwickler*innen wird davon ausgegangen, dass Software weitestgehend keine Auswirkung auf die Umwelt hat. Eine Annahme, die besonders durch die kontinuierliche Abstraktion von physischer Rechen- und Speicher-Infrastruktur und Software weiter beschleunigt wird.

Die SDIA hat im Oktober 2020 einen Fahrplan für die Realisierung einer nachhaltigen Digitalwirtschaft[31] vorgestellt, der mehr als 30 Aktivitäten umfasst, von denen sich ein Drittel mit Software beschäftigt. Die Arbeit des Umweltbundesamtes in diesem Bereich haben wir bereits mehrfach zitiert und als Beispiele aufgeführt. Wir haben ein hohes Interesse und intrinsische Motivation, die Reichweite und Wahrnehmung hinsichtlich energieeffizienter Software zu erhöhen. Unsere mehr als 60 europäischen Mitglieder unterstützen uns in diesem Vorhaben.

Kommunikationsstrategie

Die Kommunikationsstrategie für das Projekt setzt sich aus den folgenden Bestandteilen zusammen:

- *Begleitkommunikation und Aufbau einer öffentlichen, virtuellen Gemeinschaft aus Software-Entwickler*innen die sich für energieeffiziente Software interessieren und einsetzen.*
  Sie begleiten das Projekt mit eigenen Versuchen und Feedback zu den Vorgehensweisen, und stellen eine Art öffentlichen Beirat hinsichtlich der Alltagstauglichkeit der Werkzeug dar.
- *Nutzung von bestehenden Partnerschaften und Netzwerken mit einer hohen Reichweite, insbesondere in die Open Source Gemeinschaft*
  Bereits zu Beginn des Projekts laden wir Partnerorganisationen und Stiftungen ein, das Projekt zu begleiten. Hierzu zählen unter anderem: Green Web Foundation[32], Eclipse Foundation[33],

[31] SDIA Roadmap (2020): https://sdia.io/roadmap
[32] https://www.thegreenwebfoundation.org/
[33] https://www.eclipse.org/org/foundation/

Apache Foundation[34], Cloud Native Foundation[35], World Wide Web Foundation[36], RIPE NCC[37], Green IT Amsterdam[38], das European Institute for Sustainable IT und Green IT France[39].

- *Workshops mit Partnern und Repräsentanten aus der Entwicklergemeinschaft.* Die vorgeschlagenen Workshops der Ausschreibung werden wir, mit Vertretern aus den oben genannten Gruppen besetzen und auch im Projektverlauf durch eine Gemeinschaftsplattform mit kontinuierlicher Kommunikation integrieren (z.B. mit einer Slack-Gruppe[40]).
- *Publikationen zur Begleitung des Projekts.* Zusätzlich zu den gewünschten Artikeln und Veranstaltungen nutzt die SDIA ihre eigenen Kommunikationskanäle und Plattformen, um weitere Veranstaltungen, Webinare und Konferenzen zu realisieren. Dabei wird auch die weitere europäische Gemeinschaft mit einbezogen. Hinzu kommen Partnerschaften mit Publikationen wie dem Heise Verlag in Deutschland, Golem und Data Center Insider.
- *Umsetzung von virtuellen und physischen Hackathons.* Die Anforderung hinsichtlich der Veranstaltung von Hackathons unterstützen wir vollumfänglich. Jedoch werden wir zumindest den zweiten, optionalen Hackathon virtuell veranstalten, um auch der europäischen Entwickler-Gemeinschaft die Möglichkeit zu bieten teilzunehmen, einen positiven Beitrag zu leisten und an der Verbreitung von energieeffizienten Software-Entwicklungsmethoden mitzuwirken - sofern dieser zweite Hackathon vom UBA gewünscht ist.
- *Kooperation mit Tech Soup.* Tech Soup[41] ist die weltweit größte Organisation für die Distribution von Software Lösungen für gemeinnützige Organisationen und Partner der SDIA. Durch eine gemeinsame Kampagne und Implementierung einer Kennzeichnung für energieeffiziente Software können insbesondere gemeinnützige Organisationen, die sich für die Umwelt engagieren, auf umweltverträgliche Software-Lösungen hingewiesen werden und somit die Nachfrage für ebensolche erhöht werden.

[34] https://www.apache.org/
[35] https://www.cncf.io/
[36] https://webfoundation.org/
[37] https://www.ripe.net/
[38] http://www.greenitamsterdam.nl/
[39] https://www.greenit.fr/
[40] https://slack.com/intl/en-de/
[41] https://www.techsoup.org/

Durchführung der Workshops und Hackathons

Die drei Workshops zur Diskussion der Projektfortschritte werden als ganztägige Veranstaltungen abgehalten. Aufgrund der digitalen Ausrichtung der SDIA e.V. und regelmäßigen Durchführung von LinkedIn-Live Veranstaltungen wird auch die Rekomposition zu einem Online-Seminar kein Problem darstellen, sodass die Workshops unabhängig von der Corona-Situation abgehalten werden können und keine Extrakosten durch das digitale Setup entstehen.

Die SDIA e.V. übernimmt die Planung der Workshops, wozu die Vorbereitung der Sitzung, das Erstellen einer Tagesordnung und eines Ablaufschemas, sowie das Verfassen und Versenden der Einladungen gehören. Weiterhin werden die Moderation und eine ausführliche Dokumentation der Veranstaltungen übernommen. Nach den Events wird die aufbereitete Dokumentation in Form von Protokollen an die Teilnehmer übermittelt. Die neuen Erkenntnisse werden zudem in die weitere Projektausführung integriert.

Fachexperten für die geplanten Workshops werden zum großen Teil durch die eigenen qualifizierten Leute der Bietergemeinschaft organisiert. Durch das ausgeprägte Netzwerk der Bietergemeinschaft werden zusätzliche Branchenexperten und Gastredner einberufen, um die Veranstaltungen zu unterstützen. Dabei wird zum einen das Netzwerk des Öko-Institutes verwendet, das weitestgehend Kontakte aus dem Forschungsbereich aufweisen kann und zum anderen auf das pan-europäische Netzwerk der SDIA zurückgegriffen, das den europäischen Industriebereich für die Workshops abdeckt.

Ziel dieser Workshops ist

(i.) die Diskussion der formulierten Projektziele sowie die bis zum Zeitpunkt der Veranstaltung erzielten Ergebnisse mit relevanten Interessengruppen.

(ii.) die Distribution des entwickelten Wissens und Know-How der verschiedenen Akteure an das Netzwerk der Teilnehmer.

(iii.) die Vorführung und Diskussion von technische Erfolgen und Prototypen.

Neben den Workshops soll auch ein Hackathon abgehalten werden, um die Ergebnisse und korrekte Funktionsweise der entwickelten Messmethoden zu überprüfen und gegebenenfalls zu verbessern. Wie bei den Workshops stellt eine digitale Ausrichtung auch hier kein Problem dar, so dass der Hackathon unabhängig von der Pandemie-Situation durchgeführt werden kann. Die Ausführung des zweiten Hackathons als virtuelle Veranstaltung wird empfohlen, um den möglichen Teilnehmerkreis zu maximieren, bleibt aber im Angebot als optional aufgeführt.

Die SDIA e.V. übernimmt die Planung, zu der das Formulieren einer Aufgabenstellung, das Erstellen einer Webseite für den Hackathon und das Einladen geeigneter Teilnehmer gehört, sowie die organisatorische Durchführung. Die Preise werden wie in der Leistungsbeschreibung angegeben an die ersten drei Plätze vergeben. Im Anschluss an die Veranstaltung wird eine Zusammenfassung des Hackathons auf der errichteten Website veröffentlicht und mit weiteren Kommunikationsaktivitäten wie Artikeln oder Pressemitteilungen disseminiert.

Medienbegleitung durch die SDIA

Während des gesamten Projektes wird die SDIA als Medienbeauftragter fungieren, da sie durch ihr ausgeprägtes pan-europäisches Netzwerk die Verbreitung der Projektfortschritte auf die internationale Ebene ausgeweitet werden kann. Außerdem liefert die SDIA durch bestehende Partnerschaften die optimale Grundlage für alle vorgesehenen Veranstaltungen und Publikationen.

Durch das breite Netzwerk der SDIA sowie der regelmäßigen Präsenz auf branchenrelevanten Veranstaltungen werden die vier verpflichtenden Vorträge ohne Schwierigkeiten gehalten. Auch die vier Artikel sowie die Veröffentlichung in relevanten Fachmedien sind mit den internen Kenntnissen ohne Weiteres umsetzbar und werden von der SDIA bearbeitet.

Durch die internationale Ausrichtung der SDIA können vorgesehenen Artikel auf Wunsch des Umweltbundesamtes in bis zu vier Sprachen verfasst werden (Deutsch, Englisch, Französisch, Niederländisch). Dadurch wird die potentielle Reichweite für die Verbreitung der Projektfortschritte und -ergebnisse deutlich erhöht. In dem verbindlichen Angebot sind Deutsch und Englisch als Sprachen für die Veröffentlichung der geforderten vier Artikel einbezogen. Eine Veröffentlichung in Französisch oder Niederländisch wird in der Kalkulation als optional aufgeführt.

Zusätzlich bestehen durch das SDIA-Netzwerk enge Kontakte zu führenden nationalen sowie internationalen Fachzeitschriften, in denen Artikel und Publikationen sowie Berichte veröffentlicht oder beworben werden sollen. Dazu zählen unter anderem Online-Medienportale wie “Datacenter-Insider”[42], “Digital Infra Network”[43] sowie “Heise”[44] und „Golem“[45]. Auch das Öko-Institut zeichnet sich durch eine starke Medienpräsenz aus und zählt mit regelmäßigen Publikationen zu einer verlässlichen Informationsquelle auf nationaler Ebene[46], wodurch auch dieser Kanal für Publikationen verwendet wird.

[42] https://www.datacenter-insider.de/
[43] https://digitalinfranetwork.com/
[44] https://www.heise.de/
[45] https://www.golem.de/
[46] https://www.oeko.de/publikationen

Für den Fall, dass diese umfassende Medienpräsenz nicht ausreichend ist, soll außerdem eine Website entwickelt werden, auf der regelmäßig über die Fortschritte des Projektes sowie anstehende Events informiert werden soll. Auf dieser Website soll außerdem eine Datenbank errichtet werden, auf der Zugriff zu Dokumentationen sowie Publikationen zu dem Projekt gewährt wird.

Projektorganisation und Abstimmung mit dem Umweltbundesamt

Ziel dieses Arbeitspaketes ist die effiziente Koordination und Dokumentation des Projektes nach innen.

Um eine effiziente Kommunikation und Koordination zwischen den Parteien zu gewährleisten, werden regelmäßige Besprechungen durch die SDIA e.V. einberufen, um intern über den Fortschritt des Projektes zu diskutieren. Diese Konferenzen werden monatlich virtuell durchgeführt, um für einen reibungslosen Ablauf des Projektes zu garantieren.

Zwischen- und Abschlussberichte

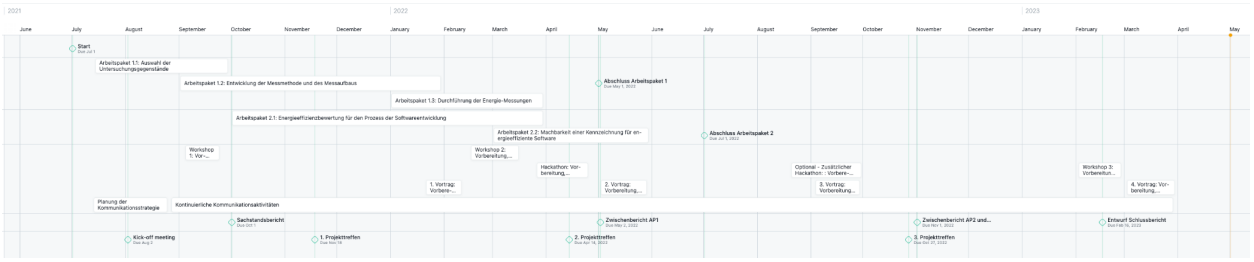
Die Berichte werden nach den Vorgaben der Ausschreibung als Entwurf vorbereitet, mit dem Umweltbundesamt abgestimmt und im Anschluss als finale Version bereitgestellt.

Projektleitung

- Projektleitung Forschung: Jens Gröger, Öko Institut e.V.
- Projektleitung Entwicklung: Flavia Paganelli, Head of Technology, Sustainable Digital Infrastructure Alliance e.V.
- Projektleitung Kommunikation: Michael Oghia, Sustainable Digital Infrastructure Alliance e.V.

Arbeits- und Meilensteinplan

Um eine zuverlässige und effiziente Durchführung des Projektes sicherzustellen, wird ein genauer Zeitplan erarbeitet, der während des Projektverlaufs einzuhalten ist. In der nachstehenden Grafik werden relevante Arbeitsschritte und Meilensteine über die Dauer des Projektes (22 Monate) aufgeführt.



Siehe Anlage 1 - Zeitplan

Kostendarstellung

Das Öko-Institut e.V. wird als Unterauftragnehmer am Projekt teilnehmen.

Von der SDIA wird sowohl ein/e Projektleiter*in als auch ein/e qualifizierte Software-Entwickler*in für das Projekt bereitgestellt - siehe Lebensläufe in den Eignungskriterien.

Die Aufstellung der Kosten finden sich in der "Anlage 2 - Kostendarstellung".

Eignungskriterien

Kriterium 1: Erfahrung in der Anwendung von Kennzahlen, Indikatoren in der Programmierung von Softwarekomponenten

Für die Einbettung der entwickelten Kennzahlen in die Software-Umgebung wird die SDIA beauftragt. Im folgenden werden drei Referenzen der für diesen Arbeitsschritt vorgesehenen Mitarbeiter Jurg van Vliet und Flavia Paganelli angegeben:

1. 30 MHz[47] (Jurg Van Vliet (Gründer u. Ehem. CEO) & Flavia Paganelli (Gründerin u. Ehem. CTO); beide SDIA)

    - Softwaregestütztes Management von Gewächshäusern (Autonomer Anbau)
    - Implementierung von Erfahrungswerten zur Bestimmung der optimalen Richtwerte
    - Big Data & Machine Learning Nutzung

[47] https://www.30mhz.com/news/autonomous-growing-is-no-longer-a-far-fetched-dream/

- Nutzung von Indikatoren und Kennzahlen wie Luftfeuchtigkeit, Lichtintensität und Temperatur, um die Bedingungen für den Anbau zu optimieren

2. ECO-Qube Projekt (2020-23)[48] (Jurg Van Vliet, SDIA)
    - KI-basiertes Kühlsystem für Rechenzentren
    - Verbindung des Kühlsystems mit IT-Lasten und elektrischer Infrastruktur
    - Ziel, die Energieeffizienz im Hinblick auf die Anforderungen an die sofortige Kühlung dynamisch zu verbessern
3. Fallstudie “From 6.2 to 0.15 seconds – an Industrial Case Study on Mobile Web Performance”[49] (Flavia Paganelli, SDIA)
    - Entwicklung eines Systems, um die Ladezeiten von Web-Applikationen zu verkürzen
    - Entwurf eines replizierbaren Performance-Engineering-Plans
    - Entwicklung eines Benchmarking Tools
    - Performance-Evaluation der Ziel-Applikation und Verbesserung der Performance

Kriterium 2: Technische Kenntnisse und Kenntnisse der Softwareprogrammierung

Auch für die Softwareentwicklung werden die eben benannte Mitarbeiter der SDIA verantwortlich sein. Im Folgenden werden die Abschlüsse, sowie die Berufserfahrung im IKT-Sektor aufgelistet. Aufgrund der aktuellen Corona-Situation werden keine beglaubigten Urkunden der Universitätsabschlüsse beigefügt. Auf Nachfrage des UBA können diese sowie andere benötigte Dokumente allerdings in Auftrag gegeben werden.

- Jurg van Vliet (Head of Product, SDIA)
    - M.Sc. Computer Science, Universität von Amsterdam
    - Autor mehrerer Bücher über AWS (Amazon Web Services) erschienen im Verlag O’Reilly[50]

[48] https://cordis.europa.eu/project/id/956059
[49] J. v. Riet, F. Paganelli and I. Malavolta, "From 6.2 to 0.15 seconds – an Industrial Case Study on Mobile Web Performance," 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME), 2020, pp. 746-755
[50] https://www.amazon.de/Jurg-van-Vliet/e/B0065SL6LI?ref=sr_ntt_srch_lnk_4&qid=1619340419&sr=8-4

- Co-Gründer von 30MHz (2014-2021)
- Co-Gründer von 9Apps (2009-2014) (AWS Beratung), darunter Kunden wie Sanoma und Usabilla

- Flavia Paganelli (Head of Technology, SDIA)
  - M.Sc. Computer Science, Universität von Buenos Aires
  - 25 Jahre Erfahrung in Softwaredesign und -development
  - Co-Gründerin und CTO bei 30 MHz (2014-2021)
  - Co-Gründerin und Cloud-Ingeneurin bei 9Apps (2009-2014)
  - Mobile Lead beim AR-StartUp Layar (2010)
  - Technische Leiterin bei TomTom (2008-2009)

Kriterium 3: Erfahrung in der Entwicklung von Kennzeichen sowie Kenntnisse über die EU-Rahmenverordnung 2017/1369 zur Festlegung der EU-Energieverbrauchskennzeichnung

Für die Entwicklung von Kennzeichen, die die Energieeffizienz von Software bestimmen sollen, wird das Öko-Institut beauftragt. Die Leitung dieser theoretischen Arbeit übernimmt Dipl.-Ing. Jens Gröger. In diesem Punkt werden vergangene sowie laufende Projekte des Öko-Instituts aufgelistet, die sich mit der Entwicklung von Kennzahlen befassen. In zwei dieser drei Referenzen übernahm Jens Gröger bereits die Leitung.

1. Identification of elements for a future "Strategy for the EU Ecolabel" [51] (Öko-Institut)
   - Verbesserung der Umsetzung des freiwilligen EU-Umweltzeichensystems
   - Strategie entwickeln, um höhere Akzeptanz für das Ecolabel erzielen

[51] https://www.oeko.de/forschung-beratung/projekte/pr-details/identification-of-elements-for-a-future-strategy-for-the-eu-ecolabel-1

2. Entwicklung und Anwendung von Bewertungsgrundlagen für ressourceneffiziente Software unter Berücksichtigung bestehender Methodik (Ende: 2018)[52] (Öko-Institut, Leitung Dipl-Ing. Jens Gröger)

    - Messung des Einflusses von Software auf den Energie- und Ressourcenverbrauch.

    - Kriterienentwicklung zur Bewertung der Umweltverträglichkeit von Software

3. 40 Jahre Blauer Engel - Weiterentwicklung seines Produktportfolios (Beginn: 2018)[53] (Öko-Institut, Leitung Dipl-Ing. Jens Gröger)

    - Entwicklung von Kriterien für die Vergabe des Umweltzeichens Blauer Engel für verschiedene Produktgruppen und Dienstleistungen

Kriterium 4: Kenntnisse in der Entwicklung von Websites

Die Website-Entwicklung wird wiederum von der SDIA übernommen. Dazu wird vor allem Flavia Paganelli die Leitung übernehmen aufgrund der hinreichenden Erfahrung im Bereich des Websitedesigns und -entwicklung.

1. Entwicklung der 30MHz Datenplattform und des Visualisierungs-Dashboards[54] , Flavia Paganelli

2. Erste Version des TomTom Web-Routenplaner (wie Google Maps)[55], Flavia Paganelli

3. Diverse Webanwendungen für unterschiedliche Kunden in Beratungsunternehmen, Flavia Paganelli

52

https://www.oeko.de/forschung-beratung/projekte/pr-details/entwicklung-und-anwendung-von-bewertungsgrundlagen-fuer-ressourceneffiziente-software-unter-beruecksic

53

https://www.oeko.de/forschung-beratung/projekte/pr-details/40-years-of-blue-angel-refinement-of-the-existing-product-range-1

54 https://www.30mhz.com/products/platform/

55 https://mydrive.tomtom.com/