Automatisierung von Messverfahren zur Bestimmung der Ressourceneffizienz von Software mittels Continuous Integration

Maik Kreutzer

Abstract: Die Informationstechnologie ist längst ein fester Bestandteil unseres Lebens sowie auch unserer Umwelt. Durch die ständige Weiterentwicklung von Produkten der Informations- und Kommunikationstechnik (IKT) sowie allgemein durch die Digitalisierung, werden die CO2-Emissionen durch den damit einhergehenden Energieverbrauch beeinflusst. Auf digitaler Ebene ist es demnach relevant, Software auf ihren Energie- und Ressourcenverbrauch zu untersuchen. Ziel dieser Publikation ist die Präsentation eines Konzepts zur automatisierten Messung der Ressourceneffizienz von Softwareprodukten, welches mit Hilfe der kontinuierlichen Integration umgesetzt wird. Als Basis wurde dafür das Messkonzept des Institut für Softwaresysteme (ISS) am Umwelt-Campus in Birkenfeld verwendet, welches bisher ohne automatisierte Messungen aufgebaut ist. Zusätzlich werden Tools vorgestellt, die für die Umsetzung und automatisierte Durchführung verschiedenster Standardnutzungsszenarien verwendet werden können.

Keywords: Green Software; Ressourceneffizienz; Automatisierung; Continuous Integration

1 Motivation und Einleitung

Eine Studie der Bitkom Akademie aus dem Jahr 2020 liefert das Ergebnis, dass digitale Technologie fast die Hälfte dazu beitragen kann, dass Deutschland seine Klimaziele bis zum Jahr 2030 erfüllt [Ak22]. Eine Voraussetzung dafür ist, dass die Energieeffizienz fortlaufend untersucht und voran gebracht wird [AE15]. Die Forschungsgruppe *Green Software Engineering*, welche zum Institut für Softwaresysteme (ISS) des Umwelt-Campus Birkenfeld gehört, beschäftigt sich mit der Energie- und Ressourceneffizienz von Software. Eines der Ziele dabei ist die Zertifizierung von nachhaltigen Softwareprodukten. Der *Blaue Engel* ist das Umweltzeichen der Bundesregierung und wurde im Jahr 2020 um Kriterien für ressourcen- und energieeffiziente Softwareprodukte erweitert [In22]. Aktuell gibt es den Blauen Engel nur für Desktop-Software, womit selbstverständlich nicht alle Arten von Softwareprodukten abgedeckt sind. Die Green Software Engineering Arbeitsgruppe plant bis Ende des Jahres 2023 auch Client-Server-Systeme und mobile Anwendungen mit dem Blauen Engel für ressourcen- und energieeffiziente Softwareprodukte zertifizieren zu können [Sc22a]. Diese Ausarbeitung entsteht in Kooperation mit dem ISS. Demnach werden durch das ISS bzw. durch die Green Software Engineering Arbeitsgruppe bekannte Modelle als Basis verwendet.

2 Grundlagen

Diese Publikation stellt ein Messkonzept vor, mit dem Softwareprodukte auf ihre Ressourceneffizienz gemessen werden können. In den Grundlagen wird erklärt, was genau überhaupt nachhaltige Software ist. An dieser Stelle wird zusätzlich erklärt, was ein Standardnutzungsszenario ist und wie dieses bei der Messung von Software helfen kann. Zudem werden die Themen Automatisierung in Verbindung mit Continuous Integration erläutert.

2.1 Nachhaltige Software und Standardnutzungsszenarien

Nachhaltige Software ist ein Teilbereich der Green IT. Darunter zu verstehen ist Software, deren direkte und indirekte negative Auswirkungen auf Menschen, Gesellschaft und Umwelt über ihren gesamten Lebenszyklus hinweg minimal sind und die bestenfalls einen zusätzlichen positiven Beitrag zur nachhaltigen Entwicklung leistet [DNH10]. Als Entwickler sollte man dementsprechend darauf achten, die Komplexität der zu entwickelnden Anwendung möglichst klein zu halten. Die Komplexität von Software ist an vielen Stellen zu erkennen, beispielsweise in der Algorithmik, der Struktur, oder dem notwendigen Software-Stack [In22]. Um ein Softwareprodukt als nachhaltig einzustufen, müssen verschiedene Kriterien erfüllt werden. Zu Beginn des Jahres 2020 hat Deutschland mit dem Blauen Engel für ressourcen- und energieeffiziente Softwareprodukte ein einheitliches Umweltzeichen für nachhaltige Software veröffentlicht [In22]. Eine Software wird dabei als besonders nachhaltig eingestuft, wenn sie die drei Kategorien des Kriterienkatalogs erfüllt. Dies betrifft die Aspekte Ressourcen- und Energieeffizienz, die potenzielle Lebensdauer der Hardware und die Benutzerunabhängigkeit [Bl22]. Zur Messung dieser Aspekte, wie beispielsweise der Ressourceneffizienz, muss für jedes Softwareprodukt ein individuelles Standardnutzungsszenario erstellt werden, welches der Beschreibung von Arbeitsabläufen und Nutzungskontext der Benutzer dient. Dabei soll eine realitätsnahe Nutzung einer Software simuliert werden. Ein solches Nutzungsszenario kann dabei ganz verschieden aussehen. Benutzer können die festgelegten Arbeitsabläufe manuell steuern, oder im besseren Fall wird ein Automatisierungstool zur Ausführung des Standardnutzungsszenarios verwendet.

2.2 Wichtigkeit von Automatisierung

Sowohl im Bezug auf nachhaltige Software als auch im Allgemeinen ist die Automatisierung von Aufgaben in der Softwareentwicklung heutzutage ein wichtiger Bestandteil des Prozesses. Werden repetitive Aufgaben automatisiert, werden Zeit und Kosten gespart. Zudem werden unnötige Fehler beim Bearbeiten von Aufgaben vermieden. Wird ein Softwaretest beispielsweise komplett manuell durchgeführt, muss der Tester genau aufpassen und darf keinen Fehler machen, da die Ergebnisse sonst verfälscht werden. Auch bei automatisierten

Tests können sich Fehler einschleichen, diese sind im Nachhinein aber i.d.R. leichter erkennbar. Ein Werkzeug, welches bei der Automatisierung im Bereich der Softwareentwicklung helfen kann, ist Continuous Integration und wird im nächsten Abschnitt genauer erläutert.

2.3 Bedeutung und Ablauf von Continuous Integration

Continuous Integration (CI) ist ein Werkzeug, welches in der Integrationsphase der Softwareentwicklung verwendet wird [Wa22]. CI wird ganz speziell zur automatisierten Integration von Codeänderungen genutzt, die mehrere Mitarbeiter an einem einzigen Softwareprojekt vornehmen. Den Entwicklern wird ermöglicht, Codeänderungen häufig an einem zentralen Platz zusammenzuführen, an dem z.B. Bauprozesse und Tests der Software ausgeführt werden [At21a]. Diese zentrale Ablage wird Repository genannt und zur Verwaltung wird i.d.R. ein Versionsverwaltungssystem verwendet. Der de facto Standard ist dabei das Tool *git* [At21b].

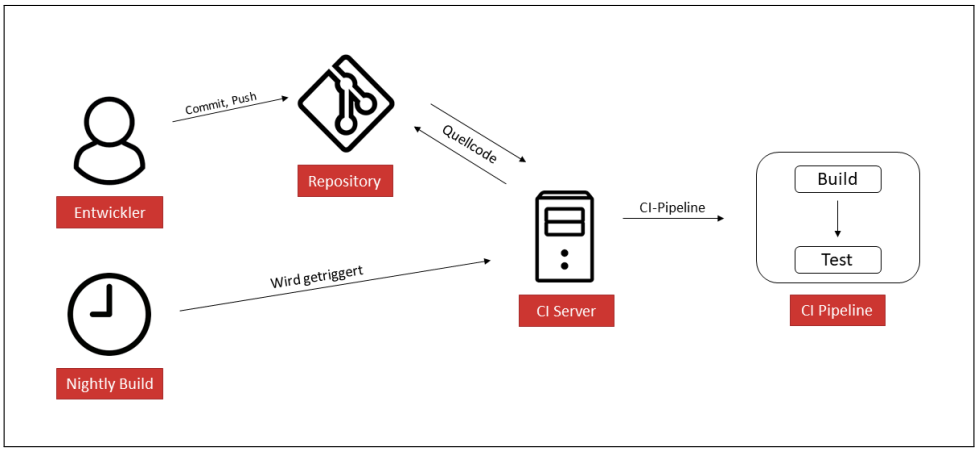

Abb. 1: Ablauf von Continuous Integration (eigene Darstellung, basierend auf [Ad22])

Abbildung 1 zeigt ein Beispiel für den Ablauf von Continuous Integration. Die erste Möglichkeit, den Prozess zu starten, erfolgt durch den Entwickler, der seine Codeänderungen mit den Operationen *Commit* und *Push* an das Repository sendet. Dies löst durch eine in dem CI-Tool konfigurierte Option automatisch die CI-Pipeline aus. Daraufhin wird der Quellcode aus dem Repository gebaut und bei Erfolg anschließend getestet. Ist der Build nicht erfolgreich, wird automatisch eine Email an den Verantwortlichen gesandt. Die andere Möglichkeit, wie sich der Prozess starten lässt, nennt sich *Nightly Build*. Die Konfiguration erfolgt ebenfalls über das CI-Tool, wobei man hier einen festen zeitlichen Start des Prozesses festlegen kann. I.d.R. wird eine Zeit wie z.B. 03:00 Uhr Nachts verwendet, da zu dieser Zeit kein Entwickler an einem Projekt arbeiten sollte.

2.4 Zwischenfazit

Die letzten Abschnitte haben geklärt was nachhaltige Software ist und wie verschiedene Softwareprodukte gemessen bzw. bewertet werden können. Zudem wurde die Bedeutung und der Ablauf des Werkzeuges Continuous Integration geklärt, welches den Prozess der Untersuchung der Ressourceneffizienz von Software unterstützen kann. Von Vorteil ist der Aspekt, dass bei jedem Build sehr einfach automatisierte Softwaretests laufen können und den Entwicklern somit regelmäßig einen Überblick über Funktionen oder Leistungen der Applikation geben. Der Begriff Continuous Integration existiert dabei schon sehr lange und die Verwendung nimmt stetig zu [Qe22]. Laut einer Studie der *International Data Corporation* aus dem Jahr 2020 setzen fast vier von fünf der befragten Unternehmen auf die kontinuierliche Integration mit samt seinen Vorteilen [ID22].

3 Messkonzept

Dieses Kapitel erläutert ausführlich das in dieser Publikation zu entwickelnde Konzept unter Betrachtung von Continuous Integration zur Messung und Evaluation von Software. Als Basis für das Messverfahren dient das bisherige Konzept der Green Software Engineering Forschungsgruppe des ISS, welches im nächsten Abschnitt vorgestellt wird.

3.1 Bisheriger Messaufbau des ISS

Das Institut für Softwaresysteme der Hochschule Trier am Standort Birkenfeld untersucht schon seit einigen Jahren den Energie- und Ressourcenverbrauch von Software. Die Forschungsgruppe Green Software Engineering des ISS entwickelt seit 2011 stetig an einer Mess- und Analysemethode zur Einschätzung des Energie- und Ressourcenaufwands anhand eines Nutzungsszenarios weiter [In22]. Die Norm ISO/IEC 14756 gilt als Anhaltspunkt für die Messung und Bewertung der Leistung sowie der Software-Effizienz von Datenverarbeitung-Systemen [IS22]. Basierend auf diesem Standard zeigt Abbildung 2 den bisherigen Messaufbau der Green Software Engineering Arbeitsgruppe.

Der Messaufbau beinhaltet die Komponenten *System Under Test (SUT)*, *Leistungsmessgerät (PM)*, *Workload Generator (WG)* sowie *Datensammlung und -analyse (DAE)*. Das Nutzungsszenario der zu messenden Software wird auf dem SUT durchgeführt. Je nach Bedarf kann ein einfacher Server, PC, Mobilgerät oder auch ein IoT-Gerät als SUT verwendet werden. Die Hardwarenutzung wird während der Ausführung des Szenarios auf dem SUT mittels der Software *Collectl* aufgezeichnet. Darunter fallen die CPU-, Disk- und RAM-Nutzung sowie der Netzdatenverkehr. Ebenfalls während dem Szenario wird die Leistungsaufnahme des SUT durch das PM gemessen. Gesteuert wird das Nutzungsszenario durch einen WG, welcher auf dem SUT eine Last erzeugt. Dies kann unter anderem durch die Ausführung eines Skripts, oder den wiederholten Aufruf einer API, Website oder Datenbank erfolgen.

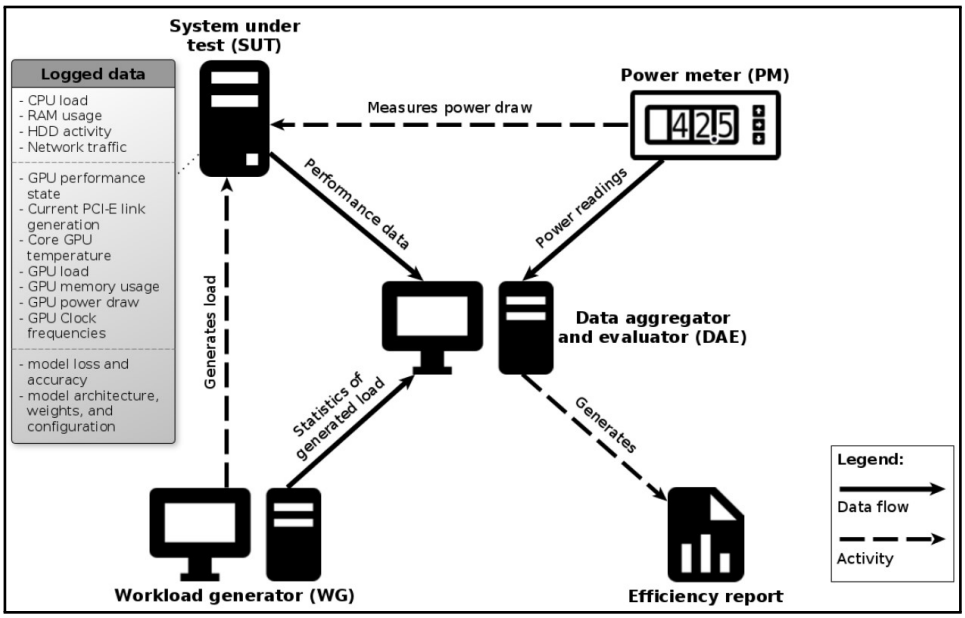

Abb. 2: Messaufbau zur Ermittlung des Energie- und Ressourcenverbrauchs [GKN21]

Ebenso ist es denkbar, dass der WG ein Werkzeug ist, welches auf dem SUT selbst ausgeführt wird. Das DAE sammelt die generierten Daten, welche während des Szenarios gemessen bzw. erstellt wurden. Darunter fallen letztendlich die Hardwarenutzung, elektrische Leistung sowie die vom WG generierte Log-Datei. Zu jeder Aktion des Szenarios wird dabei jeweils ein Start- und End-Zeitstempel aufgenommen. [In22]

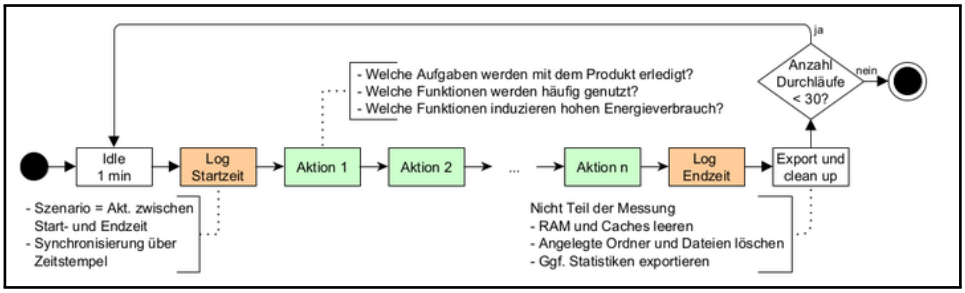

Abb. 3: Beispielhafter Ablauf des Nutzungsszenarios [In22]

Ein beispielhafter Ablauf eines Nutzungsszenarios ist in Abbildung 3 dargestellt. Um mögliche Ausreißer zu erkennen, wird das Szenario insgesamt 30 mal wiederholt. Bei jeder Messung, sei es die Ausführung von verschiedenen Funktionen der Software oder

seien es Lasttests, die die Anwendung unter erheblicher Belastung überprüfen sollen, ist es notwendig, ergänzend auch ein Baseline-Szenario zu messen. Dabei läuft die Software selbst nicht, allerdings das Betriebssystem sowie alle Voraussetzungen zur Nutzung der zu messenden Software. Der Einfluss des WG in das SUT wird somit ebenfalls in der Baseline miteinbezogen und kann später bei der Analyse herausgerechnet werden. Das SUT wird nach jeder Software auf einen Zustand vor der Installation der Anwendung zurückgesetzt, um zu gewährleisten, dass sich die Messungen nicht gegenseitig beeinflussen. [In22]

Zur Analyse der Messdaten wurde von der Green Software Engineering Arbeitsgruppe die Auswertungssoftware *OSCAR (Open Source Consumption Analysis and Reporting)* entwickelt. Dabei können sowohl Datensätze des während des Nutzungsszenarios gemessenen Energieverbrauchs als auch die entsprechende Hardware-Auslastung ausgewertet werden [Sc22b].

3.2 Messkonzept mit Continuous Integration

Basierend auf dem bisherigen Aufbau werden die Nutzungsszenarien bei dem neuen Konzept mit dem gleichen Ablauf durchgeführt. Dabei können verschiedene Automatisierungswerkzeuge verwendet werden. Für manche Werkzeuge müssen die Szenarien programmiert werden, für andere lassen sie sich ausschließlich über ein drag-and-drop-System erstellen und wieder andere kombinieren diese beiden Funktionsweisen. Eine ausführlichere Evaluation von verschiedenen Tools zur Realisierung von Standardnutzungsszenarien wird im nächsten Kapitel vorgestellt. Aus dem letzten Abschnitt ging hervor, dass auch ein Baseline-Szenario gemessen werden muss. Neben dem Standardnutzungsszenario muss demnach auch ein weiteres Szenario für die Baseline entwickelt werden.

Für das entwickelte Messverfahren ist zudem die Hardware-Auslastung besonders interessant. Bei Ausführung der Nutzungsszenarien wird darauf geachtet, die Funktionen und Module auszuführen, die besonders für einen Anstieg der Hardware-Ressourcen verantwortlich sind. Um diese Auslastung zu messen, wird unter Linux das Tool Collectl [Co18] verwendet. Das Tool zeichnet Systeminformationen zur CPU, der Platte, dem Speicher und dem Netzwerk sowie weitere Informationen jeweils mit Zeitstempel auf [Co18]. Um die Ergebnisse nach der Messung auszuwerten, gibt es bei dem Collectl-Befehl eine Option, die Daten im CSV-Format zu exportieren.

Für die Visualisierung der Ergebnisse aus den Hardware-Messungen wird für diese Ausarbeitung das Tool *Autoplot* verwendet, welches für eine Abschlussarbeit am Umwelt-Campus in Birkenfeld entwickelt wurde [Kr22]. Die Software ist in der plattformunabhängigen Programmiersprache R entwickelt. Autoplot bekommt die aus Collectl exportierte CSV-Datei übergeben und visualisiert die Messergebnisse.

	Collectl-Feld	Bedeutung
CPU	Totl	CPU-Auslastung (Prozent)
Disk	ReadKBTot / WriteKBTot	Gelesene / geschriebene Daten (KB)
Memory	Used	Verwendeter Arbeitsspeicher (MB)
Network	RxKBTot / TxKBTot	Empfangen / übertragen (KB)

Tab. 1: Relevante Collectl-Felder

Für die Auswertung relevante Collectl-Felder werden in Tabelle 1 vorgestellt. Insgesamt werden die vier Kategorien CPU, Disk, Memory und Network analysiert. Wird das Autoplot-Skript ausgeführt, wird aus der aktuell beiliegenden CSV-Datei jeweils eine Bilddatei pro Kategorie erstellt. Die erzeugten Bilder enthalten je nach Kategorie entsprechend nur einen Graphen bzw. mehrere, wenn mehr als ein Collectl-Feld für die Kategorie analysiert wird. Die Messwerte enthalten jeweils einen Zeitstempel, welcher sich auf der X-Achse aller Graphen befindet. Dargestellt werden die Stempel dabei in Sekunden-Schritten. Der Mittelwert der Datenverteilung wird durch eine rote horizontale Linie dargestellt. Eine beispielhafte Grafik zur Visualisierung der CPU-Inanspruchnahme zeigt Abbildung 4.

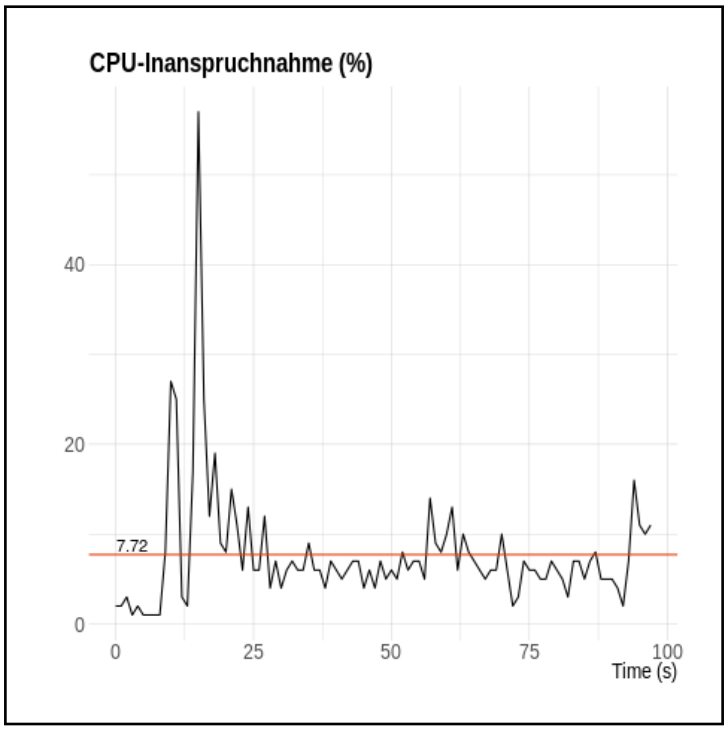

Abb. 4: Beispielhafte Visualisierung der CPU-Inanspruchnahme

3.3 Allgemeiner Ablauf

Der letzte Abschnitt hat die verschiedenen Komponenten, welche für das neue Messverfahren benötigt werden, jeweils vorgestellt. Abbildung 5 zeigt den allgemeinen Ablauf im Gesamten, um einen besseren Überblick über den Zusammenhang der Komponenten zu bekommen.

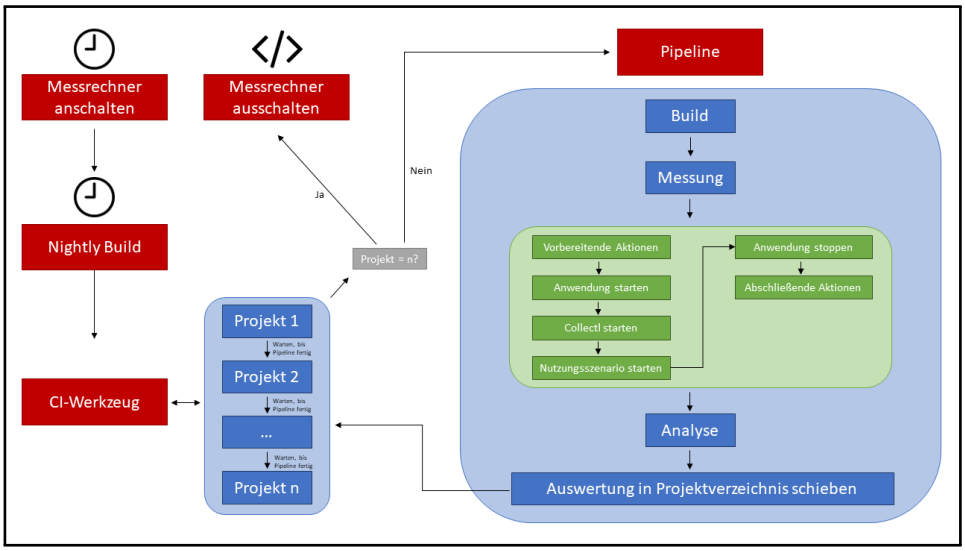

Abb. 5: Allgemeiner Ablauf des Messkonzepts (eigene Darstellung, basierend auf [GKN21])

Gestartet wird das Messverfahren durch das Einschalten des Messrechners, der hier das SUT darstellt und als CI-Server dient. Die Einschaltung erfolgt über *Wake on LAN (WOL)*, d.h. der ausgeschaltete Rechner kann über die eingebaute Netzwerkkarte gestartet werden. WOL wird zeitgesteuert über ein Skript gestartet. Der im CI-Tool konfigurierte zeitliche Auslöser startet kurz darauf den Nightly Build, d.h. die Projekte aus dem Verzeichnis werden gebaut. In jeder CI-Pipeline werden für die Projekte jeweils die gleichen Schritte durchgeführt. Zu Beginn wird das Softwareprojekt gebaut. Anschließend erfolgt die Messung, welche mehrere Schritte in einer bestimmten Reihenfolge ausführt. Zuerst müssen ggf. vorbereitende Aktionen getroffen werden, wie beispielsweise das Starten von Datenbank-Diensten. Anschließend wird die Anwendung gestartet, sowie die Hardwareaufzeichnung mit Hilfe von Collectl. Gleichzeitig wird das Nutzungsszenario gestartet. Sobald dieses fertig durchgeführt wurde, wird die Anwendung gestoppt und es werden abschließende Aktionen getroffen. Im nächsten Schritt werden Informationen zu den Hardwareressourcen an das DAE weitergeben und in diesem Fall durch das Tool Autoplot ausgewertet sowie in Form einer Grafik abgespeichert. Der Messrechner wird durch einen zeitgesteuerten Cronjob kurz nach Ablauf des letzten Projektes ausgeschaltet.

4 Evaluation von Tools zur Realisierung von Nutzungsszenarien

In den Grundlagen wurde bereits der Begriff Standardnutzungsszenario geklärt. Zur Realisierung gibt es verschiedene Tools, mit denen ein Nutzungsszenario auf automatisiertem Wege durchgeführt werden kann. Für manche Werkzeuge sind Programmierkenntnisse notwendig, für andere lassen sich die Szenarien ausschließlich über ein drag-and-drop-System erstellen. Wieder andere Werkzeuge vereinen diese beiden Funktionsweisen. In diesem Abschnitt wird für jeweils einen dieser drei Bereiche ein Tool vorgestellt, samt der jeweiligen Vor- und Nachteile.

4.1 Actiona

Mit dem Tool für Aufgabenautomatisierung Actiona lassen sich allerlei Aufgaben erstellen und durchführen. Dafür werden keine Programmierkenntnisse benötigt, die Aktionen können per drag-and-drop erstellt werden. Darunter zählen beispielweise die Emulation von Tasten- oder Mausklicks. Durch Tastenklicks können verschiedene Shortcuts angesprochen werden. Mausklicks stattdessen werden auf genauen Positionen ausgeführt, welche vorher mit Hilfe von Pixelangaben in Actiona definiert werden müssen. Ebenfalls möglich sind Systeminteraktionen wie das Öffnen eines Browsers bzw. einer URL. Genauso können auch Befehle über die Kommandozeile und zahlreiche weitere Interaktionen durchgeführt werden. Eine weitere erwähnenswerte Funktion ist die Datenmanipulation, d.h. Dateien können mit Actiona gelesen und geschrieben werden. Von Vorteil ist der Aspekt, dass sich in Actiona erstellte Dateien leicht automatisiert ausführen lassen. Zusätzlich ist die Bedienung dank des drag-and-drop-Systems sehr einfach und intuitiv. Mit dem Automatisierungstool lassen sich sowohl Web- als auch Desktopanwendungen oder sonstige Systeminteraktionen steuern. Ein großer Nachteil ist die Emulation von Mausklicks, welche zur Realisierung eines Nutzungsszenarios immer wieder benötigt werden. Diese steuern nicht die Elemente an, sondern müssen per Pixelangaben positioniert werden, was recht aufwendig ist.

4.2 Selenium

Selenium ist ein Open Source Werkzeug zur Automatisierung von Browser-Anwendungen, das seit vielen Jahren zu den bekanntesten und beliebtesten UI Testautomatisierungslösungen für Webanwendungen zählt. Unterstützt werden dabei die Browser Treiber von Mozilla, Microsoft, Google, Opera sowie Apple. Die automatisierten Tests können dabei dank der für viele unterschiedlichen Programmiersprachen bereitgestellen Selenium Language Bindings in der favorisierten Software-Entwicklungsumgebung geschrieben werden [Co09]. Die Programmiersprache Python bietet für Selenium beispielsweise eine eigene Bibliothek, welche über das Paketverwaltungsprogramm *pip* installiert werden kann. Dadurch werden dem Entwickler der Tests einige vorhandene Funktionen bereitgestellt, mit denen sich eine Webanwendung automatisiert öffnen und nach Belieben steuern lässt. Der Browser

kann beispielsweise neu geladen werden, es können verschiedene HTML- oder CSS-Elemente angesteuert werden, womit sich letztendlich auch Nutzeraktionen nachbilden lassen. Von Vorteil ist, dass im Gegensatz zum drag-and-drop Automatisierungstool Actiona mit Selenium keine Pixel angesprochen werden, sondern direkte Elemente. Zusätzlich ist man als Entwickler in einer Entwicklungsumgebung deutlich flexibler, als in Actiona. Dafür werden für Tools wie Selenium allerdings Programmierkenntnisse benötigt, unabhängig einer konkreten Programmiersprache. Für das Open Source Werkzeug gibt es zahlreiche Problemlösungen zu finden sowie auch eine gute Dokumentation, dennoch wird das Gestalten von Nutzungsszenarien mit Selenium sehr schnell kompliziert. Grund dafür kann vor allem ein komplexer Aufbau innerhalb eines Frameworks sein, welches für das Frontend verwendet wird. Mit manchen Frameworks wird es dem Entwickler erschwert, verschiedene Elemente über CSS-Selektoren anzusprechen. Dafür sind sehr fortgeschrittene Kenntnisse notwendig, weswegen man sich auch erst in den Aufbau des Document Object Model (DOM) bzw. in die Selenium-Bibliothek einarbeiten muss. Ein weiterer Nachteil im Vergleich zu Actiona ist die Beschränkung auf den Webbereich. So können zwar Nutzungsszenarien für Browser- und mobile Anwendungen erstellt werden, nicht aber zu Desktopanwendungen.

4.3 Katalon Studio

Katalon Studio ist ein Werkzeug zur Entwicklung und Durchführung automatisierter Softwaretests. Basierend auf den beiden Automatisierungs-Frameworks Selenium und Appium, bietet Katalon Studio zusätzlich eine Entwicklungsumgebung. Spezialisiert ist die Software auf die Bereiche Web-, API-, Mobile- und Desktop-Anwendungen. Szenarien können mit Katalon Studio entweder mit Hilfe von Bausteinen zusammengeklickt werden, oder per Java-Code erstellt werden. Dabei kann jederzeit zwischen den beiden Techniken gewechselt werden. Die Bedienung mit Bausteinen ist nicht komplett identisch zum drag-and-drop-System von Actiona, im Grunde lassen sich die beiden Funktionsweisen dennoch gut miteinander vergleichen. Aufgrund der Tatsache, dass Katalon Studio unter anderem auf Selenium basiert, erbt die Software auch alle Funktionen, bis auf die verfügbaren Language Bindings. Für Katalon Studio hat der Entwickler ausschließlich die Möglichkeit, Java als Programmiersprache zu verwenden. Insgesamt ist die Software also eine Vereinigung der beiden anderen Tools und somit das umfangreichste Werkzeug. Dies ist wahrscheinlich auch ein Grund, dass es für Katalon ein Preismodell gibt. Die freie Version von Katalon Studio ist die, welche in diesem Abschnitt beschrieben wurde, samt den Funktionen wie den Bausteinen und der Java-Programmierung. Die kostenpflichtigen Modelle enthalten weitere Funktionen wie beispielsweise eine direkte Integration in den CI-Prozess, welche allerdings nicht für die Realisierung eines Nutzungsszenarios benötigt werden. Mit Katalon Studio bekommen die Entwickler prinzipiell die gleichen Vorteile geboten wie auch für die beiden Tools Actiona und Selenium. Die Bedienung ist recht einfach und es lassen sich ebenfalls schnell verschiedene Szenarien basteln. Zusätzlich können Elemente direkt angesprochen werden, was zum einen keine Pixelangaben voraussetzt, zum anderen hat man aber auch hier die gleichen Nachteile wie bei Selenium. Es werden zwar für Katalon Studio keine

Programmierkenntnisse vorausgesetzt, besser ist es aber dennoch, wenn der Entwickler sich mit dem Document Object Model auskennt. Ein weiterer Nachteil der Software ist die automatisierte Ausführung der entwickelten Datei bzw. des Nutzungsszenarios, welche sich etwas komplexer gestaltet als für die beiden anderen Werkzeuge.

4.4 Empfehlung

Eine genaue Empfehlung lässt sich an dieser Stelle nicht geben. Die miteinander verglichenen Tools haben alle ihre individuellen Stärken bzw. Schwächen und unterscheiden sich im Einsatzbereich. Wen es nicht stört, dass für jeden Mausklick vorher die genauen Pixelangaben bestimmt werden müssen, wird sich unter Actiona sicherlich wohl fühlen. Jeder, der Spaß am Programmieren hat, wird auch mit Selenium ein gutes Werkzeug finden. Katalon Studio ist die Alternative für die Personen, die gerne zwischen beiden Funktionsweisen wechseln wollen. So ist es nicht immer notwendig Programmcode zu schreiben, wenn man aber an einer Stelle die gewisse Flexibilität haben möchte, kann man einfach auf Java-Code umschalten.

5 Fazit

In dieser Publikation wurde auf Basis des Messkonzepts des Institut für Softwaresysteme am Umwelt-Campus in Birkenfeld ein erweitertes Konzept entwickelt, welches mittels Continuous Integration durchgeführt werden kann. Ziel des vorgestellten Messkonzepts ist die Unterstützung aller Entwickler im Hinblick auf nachhaltige Software. Mit Hilfe der Auswertungssoftware Autoplot können verschiedene Messungen von Softwareprodukten direkt miteinander verglichen werden. Darunter zählen sowohl die Messungen zur Baseline als auch zum Standardnutzungsszenario. Ist beispielsweise die CPU-Inanspruchnahme einer Software Montags gering und die nächste Version am folgenden Tag fällt im Bezug zur CPU-Inanspruchnahme höher aus, fällt dies dank der erstellten Grafiken sofort auf. Der Vorteil hierbei ist, dass der Entwickler selbst, bis auf den initialen Aufwand, nichts machen muss und die Ergebnisse automatisiert geliefert bekommt. Durch die Messung von Softwareprodukten stehen etliche Informationen zu der Hardware-Auslastung zur Verfügung. Ein konkreter Energieverbrauch kann jedoch nicht genannt werden, da die Beziehung zwischen Energieverbrauch und Hardware-Inanspruchnahme noch nicht klar ist. Aushilfe dafür könnte eine Ergänzung des Konzepts schaffen, die neben der Messung der Hardware-Auslastung auch ebenso den Energieverbrauch automatisch misst.

Mit dem automatisierten Konzept hat man also einige Vorteile, beispielsweise regelmäßige Untersuchungen zur Ressourceneffizienz einer Software auf automatisiertem Wege, ohne die Messungen jedes Mal manuell zu starten. Dadurch wird eine Menge an Zeit gespart und Fehler werden dank automatisierter Prozesse vorgebeugt. Hingegen muss auch erwähnt werden, dass es dabei viel Zeit kostet, den initialen Aufwand zu bewältigen. Dazu zählt die

Einrichtung des CI-Servers, die zu untersuchenden Softwareprodukte müssen eingepflegt werden, wobei es dabei immer wieder zu Problemen kommen kann. Der größte Nachteil des Messkonzepts ist aktuell jedoch, dass die Werte zum Energieverbrauch komplett fehlen.

Für das neue Messkonzept wurden zusätzlich insgesamt drei Tools zur Realisierung von Nutzungsszenarien betrachtet, die alle in verschiedenen Bereichen eingesetzt werden können. Interessant wäre an dieser Stelle noch zu wissen, ob bzw. inwieweit diese Werkzeuge die Messungen zur Ressourceneffizienz beeinflussen.

Literaturverzeichnis

[Ad22] AddTeq: , jenkins_pipeline.png (PNG-Grafik, 920 × 295 Pixel), 2022.

[AE15] Andrae, Anders S. G.; Edler, Tomas: On Global Electricity Usage of Communication Technology: Trends to 2030. 6:117–157, 2015.

[Ak22] Akademie, Bitkom: , Bitkom Akademie | Wir qualifizieren die Digitalwirtschaft, 2022.

[At21a] Atlassian: , Was ist Continuous Integration? | Atlassian, 2021.

[At21b] Atlassian: , Was ist Git: Mit diesem Leitfaden wirst du zum Git-Profi, 2021.

[Bl22] Blue Angel: , Blue Angel, software products, resources and energy efficient, transparent interfaces | Blauer Engel, 2022.

[Co09] ComOrg: 3rd International Symposium on Empirical Software Engineering and Measurement. IEEE, Piscataway, NJ, 2009.

[Co18] Collectl: , collectl, 2018.

[DNH10] Dick, Markus; Naumann, Stefan; Held, Alexandra: Green Web Engineering. A Set of Principles to Support the Development and Operation of “Green” Websites and their Utilization during a Website’s Life Cycle. Filipe, Joaquim, S. 7–10, 2010.

[GKN21] Guldner, Achim; Kreten, Sandro; Naumann, Stefan: Exploration and systematic assessment of the resource efficiency of Machine Learning. INFORMATIK 2021, 2021.

[ID22] IDC: The premier global market intelligence company: , IDC: The premier global market intelligence firm, 2022.

[In22] Informatik Aktuell: , Software und Nachhaltigkeit – Wie passt das zusammen?, 2022.

[IS22] ISO: , ISO/IEC 14756:1999, 2022.

[Kr22] Kreutzer, Maik: Automatisierung von Messverfahren zur Bestimmung der Ressourceneffizienz von Software mittels Continuous Integration. 2022.

[Qe22] Qentelli: , An Introduction to Continuous Integration | Qentelli, 2022.

[Sc22a] Sciences, Hochschule Trier - Trier University of Applied: , ReFoPlan 2020, 2022.

[Sc22b] Sciences, Hochschule Trier - Trier University of Applied: , UFOPLAN-SSD 2015, 2022.

[Wa22] Wagner, Sarah: , Continuous Integration: Was es ist, warum es wichtig ist und Tools für den Einstieg, 2022.