Abschlussbericht

ContinulTy

Automatisiertes Performance-Testen in der kontinuierlichen Softwareentwicklung

Tobias Angerstein,[1] Christoph Heger,[1] André van Hoorn,[2] Dušan Okanović,[1,2] Henning Schulz,[1] Stefan Siegl,[1] Alexander Wert[1]

[1] Novatec Consulting GmbH, Leinfelden-Echterdingen
[2] Universität Stuttgart, Inst. für Software Engineering, Stuttgart

Das diesem Bericht zugrunde liegende Vorhaben wurde mit Mitteln des Bundesministeriums für Bildung und Forschung unter dem Förderkennzeichen 01IS17010 gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt bei den Autoren.

Inhaltsverzeichnis

1 Einleitung

Viele Unternehmen haben bereits den Wandel zu einem Technologieunternehmen vollzogen, in dessen neuem Mittelpunkt der Wertschöpfungskette ein Anwendungssystem steht. Die Nutzer dieses Anwendungssystems haben in der heutigen schnelllebigen Welt ein gesteigertes Qualitätsbewusstsein. Anwendungssysteme, die durch die Nutzer als langsam und reaktionsträge wahrgenommen werden, werden nicht länger toleriert. Die vorherrschende Konkurrenzsituation macht es den Nutzern leicht, zwischen verschiedenen Angeboten zu wählen und zu wechseln. Haben sich die Nutzer erst für das Angebot eines Wettbewerbers entschieden, geben sie dem abgekehrten Anwendungssystem in den seltensten Fällen eine zweite Chance. Der Unternehmenserfolg wird dadurch nachhaltig geschädigt. Um Performance-Probleme deshalb frühzeitig bereits während der Implementierung von Änderungen zu erkennen, steigen die Anforderungen an das Performance-Testen, um die Qualität des Anwendungssystems sicherzustellen. Eine wesentliche Form der Performance-Tests sind Lasttests [JH15], in denen das System in einer Testumgebung synthetischen Anfragen ausgesetzt wird, deren Eigenschaften repräsentativ für das Nutzungsprofil (Lastintensität, Verhalten der einzelnen Benutzer, Eingabedaten) der realen Betriebsumgebung sind.

Problem 1: Es existieren keine oder schlechte Lasttests. *Die Praxis zeigt, dass Lasttests häufig gar nicht oder in einer schlechten Qualität vorhanden sind bzw. durchgeführt werden. Typische Gründe dafür sind der benötigte Aufwand und die benötigte Kompetenz zur Erstellung repräsentativer Lasttestskripte.*

Problem 2: Lasttests veralten und sind aufwendig zu warten. *Lasttestskripte veralten, sobald sich technische Schnittstellen oder das Nutzungsprofil ändern. Dies führt auch dazu, dass Lasttests bedingt durch hohen Zeitdruck zur Auslieferung häufig ausgelassen werden, da der Aufwand zur Wiedererreichung der Lauffähigkeit zu hoch ist.*

Verschärft wird die Problematik durch moderne Softwareentwicklungsparadigmen. Die kontinuierliche Softwareentwicklung [Bo14, BWZ15] bricht mit der klassischen Rollenaufteilung zwischen Entwicklung und Betrieb.[1] Die interdisziplinären Teams (DevOps-Teams) sind sowohl für die Entwicklung als auch den Betrieb eines Anwendungssystems verantwortlich. Durch diese organisatorische Änderung und einen hohen Grad an Automatisierung können heutzutage deutlich häufiger (bis zu mehrmals am Tag) und schneller neue Versionen eines Anwendungssystems in Betrieb genommen werden. Dies ermöglicht Unternehmen, schnell und flexibel auf Kundenbedürfnisse reagieren zu können. Die schnelle Reaktionsmöglichkeit steigert auch die Konkurrenzfähigkeit des Unternehmens und das Wertangebot für den Kunden. Um das Potential voll ausschöpfen zu können, müssen die Anwendungssysteme in einem geeigneten Architekturstil entwickelt werden. Der Microservice-Architekturstil [Ne15] hat sich für diese Aufgabe in der Industrie bewährt.

[1] Insbesondere in der Industrie ist der Begriff DevOps gängig. Im vorliegenden Bericht werden diese Begriffe synonym verwendet.

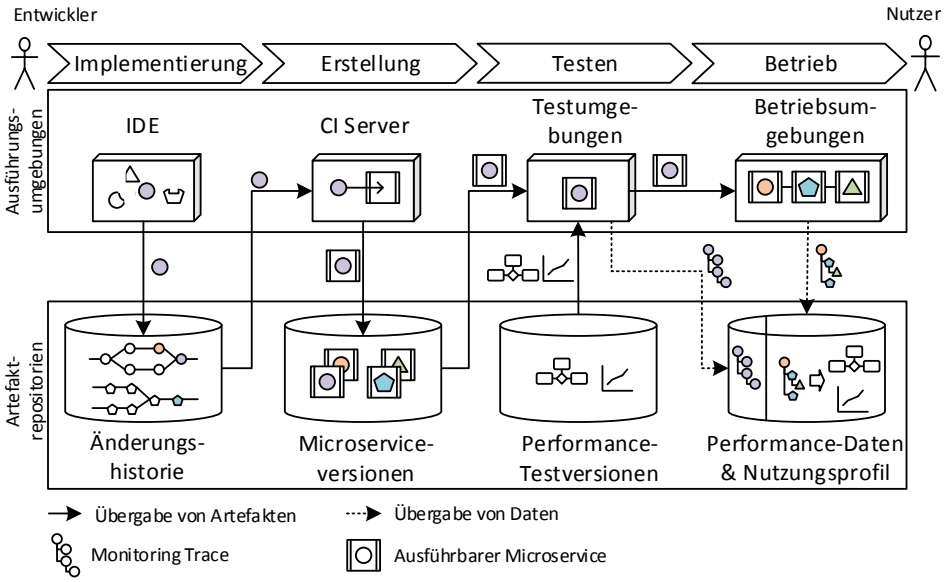

Abbildung 1: Schematische Darstellung des Prozesses von der Entwicklung bis zum Betrieb eines Dienstes

Abbildung 1 stellt den Prozess von der Entwicklung bis hin zum produktiven Betrieb eines Microservices im Rahmen der kontinuierlichen Softwareentwicklung schematisch dar. Dabei werden die Stufen Implementierung, Erstellung, Testen und Betrieb der kontinuierlichen Softwareentwicklung durchlaufen. Entwickler nehmen Änderungen an der Implementierung des Dienstes vor, um beispielsweise neue Funktionalitäten hinzuzufügen oder Fehler zu beheben. Die Änderungen werden zur Änderungshistorie hinzugefügt (Commit), die von einem Versionsverwaltungssystem (z. B. Git, SVN) verwaltet werden. Die Änderungen werden durch den Continuous-Integration-Server (z. B. Jenkins) zu einem ausführbaren Dienst zusammengebaut (z. B. Docker-Container). Die neue Version des Dienstes wird in einem Artefaktrepositorium gespeichert (z. B. Sonatype Nexus), in dem auch frühere Versionen verwaltet werden. Die neue Version des Dienstes wird auf einer oder mehreren Testumgebungen zur Ausführung bereitgestellt und durch funktionale und nicht-funktionale Tests zur Qualitätssicherung überprüft. Dazu gehört auch die Ausführung von Performance-Tests. Der Lasttest als eine Art von Performance-Test simuliert eine Menge von Nutzern, die mit dem Dienst interagieren gemäß dem Nutzungsprofil, das im Betrieb beobachtet wird. Die mit Hilfe von Monitoring-Werkzeugen gesammelten Daten werden gespeichert. Wenn alle Anforderungen erfüllt sind, wird die neue Version in Betrieb genommen. Im laufenden Betrieb werden dann kontinuierlich weitere Monitoring-Daten gesammelt.

Problem 3: Nicht für jede Änderung kann ein vollständiger Lasttest durchgeführt werden. *Lasttests beanspruchen in der Regel mehr Zeit als reine funktionale Tests. Bedingt durch die hohe Frequenz mit der neue Versionen eines Dienstes in Betrieb genommen werden, kann nicht jede Version eines Dienstes durch einen vollständigen Lasttest überprüft werden. Beispielweise wird bei Amazon, Flickr und Etsy jede erfolgreiche Änderung in Dienst gestellt, was zu Hunderten von Änderungen im Betrieb pro Tag führen kann.*

Problem 4: Aufwendige Diagnose von Performance-Regressionen bei aggregierten Änderungen. *Wird ein Lasttest ausgeführt, so werden in einer Version meist auch zuvor nicht getestete Änderungen früherer Versionen mitgetestet. Wird ein Performance-Problem im Test oder in der Produktion erkannt, muss aus den aggregierten Änderungen die Änderung isoliert werden, die für das Problem verantwortlich ist. Häufig sind jedoch auch Änderungen bzw. nicht bedachte Eigenschaften im Benutzungsprofil verantwortlich.*

Das Ziel des Projekts *ContinuITy* war es, die genannten Problemstellungen zu adressieren. Die genaue Aufgabenstellung wird im folgenden Kapitel erläutert. Es folgt eine Übersicht über die erzielten Ergebnisse und Erfahrungen.

2 Kurzdarstellung

2.1 Aufgabenstellung

Das Projekt *ContinuITy* adressiert die zuvor genannten Problemstellungen durch eine innovative Lösung zum automatisierten Performance-Testen — eingebettet in Prozesse und Infrastruktur der kontinuierlichen Softwareentwicklung. Ziel ist durch Ausnutzung von kontinuierlich aufgezeichneten Messdaten aus dem Produktivbetrieb, automatisiert effizientes und nachhaltiges Lasttesten zu gewährleisten und in die kontinuierliche Softwareentwicklung zu integrieren. Lasttests werden automatisiert aus Messdaten extrahiert und evolviert. Eine modulare Beschreibungssprache ermöglicht die Definition von Lasttests, die durch zusätzliche Semantik — z. B. über Testart und -ziele — angereichert werden können. Im Rahmen der Automatisierung des Softwareerstellungsprozesses (Continuous Delivery) erfolgt eine Auswahl relevanter Lasttests, die Erkennung von Regressionen und deren Diagnose.

Die in Abschnitt 1 aufgeführten Probleme werden durch die folgenden Eigenschaften adressiert, welche auf den Prozessen und den Infrastrukturender kontinuierlichen Softwareentwicklung basieren:

Automatische Extraktion und Wartung von Lasttests Aus den von APM-Werkzeugen kontinuierlich gesammelten Messdaten werden automatisiert Modelle des Nutzungsprofils extrahiert und analysiert sowie Lasttests extrahiert. Die Erzeugung der Lasttests aus den Nutzungsprofilen des Produktionssystems stellt sicher, dass diese Lasttests zu jeder

Zeit repräsentativ und ausführbar sind. Die Evolution des Anwendungssystems und des Nutzungsprofils wird hierbei explizit berücksichtigt. Ferner gehen manuelle Anpassungen der Lasttests nicht verloren, sondern werden explizit bei der Aktualisierung einbezogen. Die Extraktion beinhaltet nicht nur die Lasttestskripte, sondern auch die Ausführungsumgebung, die mittels moderner Technologien wie Container-basierter Virtualisierung erstellt wird. Auch wird das Verhalten anderer Microservices auf Basis der Daten aus der Produktivumgebung emuliert.

Modularisierte und semantifizierte Lasttests Zur Definition der Lasttests bietet *ContinuITy* eine modulare Beschreibungssprache. Diese erlaubt die Trennung von automatisch extrahierten Anteilen und manuellen Anpassungen, die Wiederverwendung von wiederkehrenden Elementen der Lasttests sowie die Separierung von Lasttests in Lasttestsuits, die aus einzelnen Lasttests besteht, welche jeweils bestimmte Belange bedienen. Eine zusätzliche Semantifizierung erlaubt das Annotieren der Teillasttests mit zusätzlichen Informationen wie Nutzungsprofile mit wichtigen fachlichen Anfragen, häufige Benutzeraktionen, Lasttests für bestimmte Tageszeiten mit geringer/hoher Last oder langlaufende Tests zur Erkennung von potentiellen schleichenden Performanzdegradierungen sowie der Dauer der Lasttests.

Erkennung von Performance-Regressionen durch Lasttests Die Lasttests werden innerhalb der Continuous-Delivery-Pipeline eingesetzt, um Performance-Regressionen vor der Auslieferung der Software in die Betriebsumgebung zu erkennen. Lasttestumgebungen werden automatisiert provisioniert und zur Ausführung von Lasttests genutzt. Die Ergebnisse der Lasttests werden automatisiert ausgewertet und analysiert, um Performance-Regressionen auf Basis der während der Tests gesammelten Messdaten zu erkennen. Die Semantifizierung der Lasttests wird von *ContinuITy* genutzt, um basierend auf der Häufigkeit von Releases, der Verfügbarkeit von Testumgebungen sowie der Dauer, aber auch der Relevanz der Lasttests zu entscheiden, welche Lasttests für eine jeweilige Änderung ausgeführt werden.

Automatisierte Diagnose von Performance-Regressionen *ContinuITy* untersucht die Ursache von Regressionen. Wird eine Regression erkannt, analysiert *ContinuITy* zunächst, ob diese durch eine Änderung des Nutzungsprofils oder der Implementierung bedingt ist. Die Analyse erfolgt auf Basis der Änderungshistorie sowie der Information, welche Lasttests vor der Regression (nicht) ausgeführt wurden. Das Ergebnis wird aufbereitet und dem DevOps-Team zur Verfügung gestellt, um anschließend durch Änderungen die Problemursache zu beheben. Diese Änderungen werden wiederum automatisiert durch das in diesem Abschnitt detaillierte Verfahren getestet.

2.2 Voraussetzungen

Um die Diagnose von Performance-Problemen zu automatisieren, arbeiteten die Universität Stuttgart und die Novatec bereits im Forschungsprojekt diagnoseIT zusammen [He16, He18], welches ebenfalls über die Initiative KMU-innovativ gefördert wurde. Trotz des anderen thematischen Kontexts, sollten Projektergebnisse aus diagnoseIT in *ContinuITy* verwendet und ggf. weiterentwickelt werden. Hierzu zählen u. a. das werkzeugunabhängige Format OPEN.xtrace [Ok16] zur Trace-Repräsentation sowie der Ansatz zur Diagnose von Performance-Problemen. Darüber hinaus verfügen die Antragsteller über zahlreiche (auch gemeinsame) Vorarbeiten, auf die in *ContinuITy* aufgebaut werden soll. Besonders hervorzuheben ist der WESSBAS-Ansatz [Vö16].

2.3 Planung und Ablauf

Die Laufzeit des Projekts *ContinuITy* erstreckte sich von September 2017 bis Februar 2020.

Das Projekt wurde in die folgenden sechs Arbeitspakete (AP 1–6) strukturiert. Die folgende Beschreibung der Arbeitspakete erfolgt aus der Perspektive der Vorhabenbeschreibung.

- *AP 1 (Automatische Extraktion und Wartung von Lasttests).* In diesem Arbeitspaket werden die Verfahren entwickelt, um Lasttests automatisiert aus vorhandenen Monitoring-Daten aus dem Produktivbetrieb zu erzeugen und aktuell zu halten. Die Lasttests werden in dem dafür vorgesehenen Repositorium abgelegt. Zunächst erfolgt die Definition eines Werkzeug-unabhängigen Datenformates für APM-Messdaten. Hier ist geplant, auf dem bereits verfügbaren Datenformat OPEN.xtrace [Ok16] aufzubauen und notwendige Erweiterungen vorzunehmen. Mit Hilfe des Datenformates erfolgt anschließend die Anbindung von APM-Werkzeugen, um Monitoring-Daten aus dem Produktivbetrieb abzugreifen. Es wird eine Werkzeug-unabhängige Beschreibungssprache für die Repräsentation von Nutzungsprofilen, definiert, deren Instanzen automatisiert aus Monitoring-Daten extrahiert werden. Aus den Instanzen der Beschreibungssprache findet anschließend die automatisierte Generierung von Lasttests für ausgewählte Werkzeuge statt. Als Grundlage für die Beschreibungssprache und die Extraktion der Instanzen wird auf WESSBAS [Vö16] aufgebaut.
- *AP 2 (Modularisierung und Semantifizierung von Lasttests).* In diesem Arbeitspaket findet eine Strukturierung von Lasttests statt, um eine lose Kopplung automatisch extrahierter und manuell angepasster Anteile zu erreichen. Außerdem werden durch Annotationen die Fachlichkeit, Testziele sowie Testeigenschaften eines Lasttests ausgedrückt. Hierzu wird zunächst die Beschreibungssprache um die benötigten Konstrukte für die Modularisierung und Semantifizierung erweitert. Darüber hinaus wird die Extraktion dahingehend erweitert, dass die Modularisierung und Semantifizierung (semi-)automatisiert vorgenommen wird.

- *AP 3 (Kontinuierliches Performance-Regressionstesten).* In diesem Arbeitspaket erfolgt die Integration der in AP 1und AP 2 entwickelten Ansätze in die Infrastruktur der kontinuierlichen Softwareentwicklung. Zunächst erfolgt die Realisierung der automatisierten Ausführung von Lasttests und das Sammeln von Monitoring-Daten der Testausführung und das Persistieren der Daten in dem dafür vorgesehenen Repositorium. Aufbauend auf den gesammelten Monitoring-Daten findet eine Erkennung von Performance-Regressionen statt. Schließlich erfolgt die automatisierte Selektion und Komposition der Lasttests basierend auf dem definierten Testziel.

- *AP 4 (Automatisierte Diagnose von Regressionen).* In diesem Arbeitspaket erfolgt die Diagnose, welche Änderung(en) zur erkannten Performance-Regression geführt haben sowie die Berechnung und Aufbereitung der Regressionsauswirkungen auf die Performance in einer Form, dass sie von Entwicklern verstanden werden kann. Zunächst erfolgt die Diagnose des Nutzungsprofils, das bspw. Aufschluss darüber gibt, ob eine gesteigerte Intensität, neue Eingabedaten oder ein geändertes Nutzerverhalten (z. B. durch ein neues Serviceangebot) zur Performance-Regression geführt hat. Außerdem erfolgt die Diagnose der Implementierungsänderungen, die Aufschluss darüber gibt, welche Teile der Anwendung modifiziert wurden, ob die Softwarekonfiguration geändert wurde und ob Bibliotheken ausgetauscht wurden. Schließlich erfolgt die Aufbereitung der Auswirkungen, bspw. dass der Austausch einer Bibliothek zur Verdopplung der Datenbankabfragen geführt hat.

- *AP 5 (Validierung und Evaluation von* ContinuITy*).* In diesem Arbeitspaket erfolgt die Bewertung der in den inhaltlichen Arbeitspaketen (AP 1–4) entwickelten Ansätze hinsichtlich qualitativer und quantitativer Fragestellungen. Hierzu zählen alle Tätigkeiten, die für eine prototypische Umsetzung der Konzepte notwendig sind sowie die Durchführung von Laborstudien und Fallstudien.

  Konkrete Forschungsfragen sind den jeweiligen Arbeitspaketen zugeordnet und betreffen beispielsweise den Automatisierungsgrad der Lasttestextraktion, die Qualität der extrahierten Lasttests, den Mehrwert der Modularisierung und Semantifizierung sowie die Qualität der Regressionserkennung (z. B. Sensitivität und Spezifizität der Klassifikation), Lasttestselektion (z. B. eingesparte Testzeit vs. Testabdeckung) und Diagnose (z. B. Genauigkeit der analysierten Ursache und Auswirkungen).

  Ein wichtiger zu untersuchender Aspekt, der sich über alle Arbeitspakete erstreckt, ist darüber hinaus die Skalierbarkeit der entwickelten Konzepte und Werkzeuge — u.a. in Bezug auf Systemgrößen und Lastszenarien sowie inwieweit die in *ContinuITy* angestrebte Automatisierung erreicht wurde.

- *AP 6 (Projektkoordination und Dissemination).* Dieses Arbeitspaket umfasst alle Tätigkeiten, die mit der i) Projektkoordination — d. h. Organisation, Erfolgskontrolle, Steuerung und Berichterstattung (Zwischenberichte und Abschlussbericht) des Projektes verbunden sind sowie solche, ii) die dazu dienen, die in *ContinuITy*

erarbeiteten Forschungsergebnisse während der Projektdurchführung einer breiteren Öffentlichkeit zur Verfügung zu stellen (Dissemination).

2.4 Wissenschaftlicher und technischer Stand vor Projektdurchführung

Die zu *ContinuITy* bezüglich des Stands der Wissenschaft und Technik verwandten und in diesem Abschnitt behandelten Bereiche sind Software Performance Engineering und Application Performance Management, Performance- und Lasttesten sowie Erkennung und Diagnose von Performance-Regressionen. Die wesentlichen Eigenschaften der kontinuierlichen Softwareentwicklung wurden im Abschnitt 1 genannt.

2.4.1 Software Performance Engineering und Application Performance Management

Das Forschungsfeld Software Performance Engineering umfasst sämtliche Aktivitäten im Software-Lebenszyklus, die der Sicherstellung und Analyse von Performance-Anforderungen dienen [WFP07]. Es lassen sich modellbasierte Ansätze (z. B. Performance-Vorhersage [Ba04, Ko10]) und messbasierte Ansätze (z. B. Workload-Charakterisierung [CMT16] und Lasttesten [JH15]) unterscheiden. Vor allem in der Industrie herrschen messbasierte Ansätze zur Performance-Analyse vor. Hier ist der Begriff Application Performance Management (APM) geläufig. Es existieren zahlreiche kommerzielle [HCS15] (z. B. AppDynamics, Dynatrace, CA APM) und Open-Source-APM-Werkzeuge mit Fokus auf dem kontinuierlichen Monitoring von Anwendungssystemen im Produktivbetrieb, d. h. dem Sammeln und Auswerten von Performance-relevanten Messdaten wie Traces und Ressourcenauslastung. Die Antragsteller treiben seit fast 10 Jahren die Entwicklung der Open-Source-Werkzeuge inspectIT[2] [BS11] und Kieker[3] [vHWH12, Ho09] maßgeblich voran. So existieren beispielsweise zahlreiche Ansätze, um Performance-Modelle aus APM-Daten zu extrahieren oder Modelle zur Laufzeit zu verwenden, um Anwendungssysteme bezüglich ihrer Performance zu optimieren [Br15].

Die Fragestellung, wie SPE/APM in Prozesse und Infrastrukturen der kontinuierlichen Softwareentwicklung integriert werden können, ist ein Thema, das erst seit Kurzem sowohl in der Industrie als auch der Forschung zunehmend an Interesse und Relevanz gewinnt. Ein Indikator sind entsprechende Veranstaltungen wie die Workshop-Serie zu Quality-Aware DevOps[4] und das GI-Dagstuhl-Seminar zu „Software Performance Engineering in the DevOps World"[5] sowie die SPEC Research Working Group zu DevOps Performance,[6] die

[2] http://www.inspectit.eu/

[3] http://kieker-monitoring.net/

[4] QUDOS@ESEC/FSE '15 und QUDOS@ISSTA '16: http://qudos2015.fortiss.org/, http://qudos2016.fortiss.org/

[5] http://www.dagstuhl.de/16394

[6] https://research.spec.org/devopswg/

speziell diese Fragestellung adressieren und an denen die Antragsteller in führender Rolle beteiligt sind.

2.4.2 Performance- und Lasttesten

Performance-Tests können, wie auch funktionale Tests [Li09], auf unterschiedlichen Implementierungsebenen während der Entwicklung durchgeführt werden (z. B. Unit-Tests, Komponententests, Integrationstests, Systemtests und Akzeptanztests).

Bei Performance-Unit-Tests werden über Annotationen im Code Nebenläufigkeit und Anforderungen an die Ausführungszeit spezifiziert und analysiert. Hierfür stehen Werkzeuge zur Verfügung (z. B. ContiPerf [Be15], JUnitPerf [Cl10], JPerf [GG15] oder ScalaMeter [Pr16]), die i. d. R. Erweiterungen von funktionalen Unit-Test-Werkzeugen sind. Forschungsarbeiten [HHF13, Ho13] und Projektinaktivitäten (z. B. anhand der Werkzeuge ContiPerf, JUnitPerf) zeigen, dass sich Performance-Unit-Testen in den vergangenen zehn Jahren in der Praxis nicht durchgesetzt hat.

Lasttests [JH15] zielen hingegen darauf ab, Anwendungssysteme (oder Teilkomponenten wie Microservices) über definierte Schnittstellen (z. B. HTTP/REST) definierten Nutzungsprofilen in Form von nebenläufigen Anfragen auszusetzen und entsprechende Messdaten über das Performance-Verhalten des Systems für diese Lastszenarien zu sammeln. Zahlreiche kommerzielle (z. B. Borland Silk Performer und HP LoadRunner) und Open-Source-Lasttestwerkzeuge (Apache JMeter, Gatling) stehen zur Verfügung. In der industriellen Praxis wird die Last i. d. R. in Form von (deterministischen) Skripten definiert, die entweder manuell erstellt wurden oder mit Hilfe von Skriptrekordern aufgezeichnet werden. In der Forschung sind darüber hinaus probabilistische analytische Modelle wie Markov-Ketten bekannt. Die Antragsteller entwickeln beispielsweise den WESSBAS-Ansatz [Vö16], der die Werkzeug-unabhängige Definition und Extraktion von probabilistischen Nutzungsprofilen und die Generierung von Skripten für Lasttestwerkzeuge (und modellbasierte Performance-Vorhersage) erlaubt.

2.4.3 Erkennung und Diagnose von Performance-Regressionen

Unabhängig von der Art der Performance-Tests werden diversifizierende Regressionstests [Li09] dazu eingesetzt, um die Performance-Eigenschaften verschiedener Software-Versionen miteinander zu vergleichen. Das Ziel ist dabei die Erkennung und anschließende Diagnose (Ermittlung der Ursache) eines unerwünschten negativen Effektes auf die Performance.

Die Erkennung von Regressionen basiert häufig auf der Analyse von Metriken wie Ausführungszeiten, Anzahl Datenbankaufrufe, Anzahl Web-Service-Aufrufe, Anzahl Exceptions

sowie die Ressourcenauslastung von passiven Ressourcen (z. B. Warteschlangen, Paketgrößen) und aktiven Ressourcen (z. B. Speicher, I/O, und CPU) [RK16, Be14]. Bei den eingesetzten Algorithmen handelt sich um etablierte statistische Verfahren (z. B. Anomalieerkennung, Zeitreihenanalyse, Hypothesentest), die auch in vielen anderen Bereichen eingesetzt werden. Die Herausforderung bei der Regressionserkennung ist, dass bei einer zu pessimistischen Einstellung zu viele falsch-positive Ergebnisse (Fehlalarme) generiert werden und dass bei einer zu optimistischen Einstellung zu viele falsch-negative Ergebnisse — d. h. das Verfahren schlägt nicht an — generiert werden. Eine weitere Kategorie von Werkzeugen kann als Plugin in die Continuous-Integration integriert werden (z. B. Jenkins Performance Plugin). Diese bieten i. d. R. einfache statische (relative) Schwellwerte.

Ansätze zur Diagnose von Performance-Regressionen basieren häufig auf der Änderungshistorie in modernen Versionsverwaltungsystemen [ABV16], welche einen gerichteten azyklischen Graphen darstellen. Zur Identifikation der für eine Performance-Regression verantwortlichen Änderung eignen sich deshalb grundsätzlich Graphensuchverfahren für diesen Graphentyp. Ein wichtiges Kriterium bei der Auswahl ist die Effizienz des Graphensuchverfahrens, um möglichst wenige Versionen testen zu müssen. In der Praxis nutzt beispielsweise das Versionsverwaltungsystem Git einen Bisection-Algorithmus. Dieser Algorithmus wurde bereits in unseren Vorarbeiten eingesetzt [HHF13].

2.5 Zusammenarbeit mit anderen Stellen

Während der Projektlaufzeit wurde intensiv mit den assoziierten Partnern zusammengearbeitet. Wie in der Beschreibung der Ergebnisse genannt, erfolgte ferner eine Zusammenarbeit mit weiteren wissenschaftlichen Partnern (alphabetisch): Charles-Universität Prag (Tschechien), Concordia-Universität (Kanada), Gran Sasso Science Institute (Italien), Universität Alberta (Kanada), Universität Bozen-Bolzano (Italien), Universität Duisburg-Essen, Universität Kiel, Universität Lugano (Schweiz), Universität Rio de Janeiro (Brasilien) und Universität Würzburg. Einige der Kooperationen wurden im Kontext der Research Group der Standard Performance Evaluation Corporation (SPEC) durchgeführt.

3 Eingehende Darstellung

Im Folgenden wird im Detail auf die wesentlichen im Projekt *ContinuITy* durchgeführten Arbeiten und Ergebnisse, die Mittelverwendung, den Bezug zu den Förderzielen, sowie die Veröffentlichungen eingegangen.

3.1 Zusammenfassung der Ergebnisse

Die folgende Darstellung der Ergebnisse orientiert sich an den in Abschnitt 2.3 genannten Arbeitspaketen.

3.1.1 AP 1: Automatische Extraktion und Wartung von Lasttests

Ziel des Arbeitspakets war die Entwicklung von Verfahren, um Lasttests automatisiert von vorhandenen Monitoring-Daten aus dem Produktivbetrieb zu erzeugen und aktuell zu halten.

Die wesentlichen Aktivitäten und Ergebnisse sind:

- Als werkzeugunabhängige Beschreibungssprache für Nutzungsprofile wurde die WESSBAS-DSL [Vö16] wiederverwendet und angepasst. Die von WESSBAS bereitgestellte Generierung von Nutzungsprofilen und anschließende Generierung von Apache-JMeter-Lasttests wurde ebenfalls wiederverwendet und erweitert.
- Basierend auf der in AP 2 um einen lose gekoppelten Teil manueller Eingaben erweiterte WESSBAS-DSL wurde ein Ansatz zur semi-automatisierten Evolution der manuellen Eingaben entwickelt. Dadurch konnte der manuelle Wartungsaufwand erheblich reduziert werden. Die Ergebnisse wurden publiziert [SvHW20]
- Als werkzeugunabhängiges Eingabeformat von Messdaten wurde OPEN.xtrace [Ok16] verwendet und eine Konvertierung in das für WESSBAS erforderliche Eingabeformat entwickelt. Das OPEN.xtrace-Format wurde um eine Serialisierung in JavaScript Object Notation (JSON) erweitert. Dadurch kann das Format zum einfachen Austausch in der prototypischen Anwendung verwendet werden. Des Weiteren wurde OPEN.xtrace um Elemente erweitert, die das bekannte OpenTracing[7] abbilden.
- Das Lasttest-Werkzeug BenchFlow [FP18] wurde in den Ansatz integriert. BenchFlow ist ein wissenschaftlicher Ansatz zur deklarativen Beschreibung und automatisierten Ausführung von Lasttests, der im Vergleich zum bisher verwendeten Werkzeug JMeter Vorteile bezüglich Nutzbarkeit und Automatisierung mit sich bringt. Es können nun auch BenchFlow-Lasttests generiert und automatisiert ausgeführt werden. Das Konzept wurde in Zusammenarbeit mit dem als Gastwissenschaftler im Projekt tätigen Herrn Vincenzo Ferme erarbeitet. Dazu gehört eine Transformation von WESSBAS-Lastmodellen in BenchFlow-Lasttests.

3.1.2 AP 2: Modularisierung und Semantifizierung von Lasttests

Ziel des Arbeitspakets war die Strukturierung von Lasttests, um eine lose Kopplung automatisch extrahierter und manuell angepasster Anteile zu erreichen.

[7] `http://opentracing.io/`

Die wesentlichen Aktivitäten und Ergebnisse sind:

- Die WESSBAS-DSL wurde um manuelle Anpassungen wie die Spezifikation von Eingabedaten erweitert. Dazu wurde eine weitere Beschreibungssprache, die die WESSBAS-DSL ergänzt, entwickelt. Die separate Beschreibungssprache trennt manuell erstellte Modelle von den automatisch generierten WESSBAS-DSL-Instanzen. Dadurch wird die Wartbarkeit erhöht. Der Ansatz ist Teil der Publikation [SvHW20].
- Es wurden ein Ansatz zur Modularisierung von Lasttests für Microservices erarbeitet. Dadurch können Lasttests für einzelne Microservices generiert werden, um das benötigte Deployment und somit die benötigten Ressourcen zu verringern. Der Ansatz erweitert den verwendeten WESSBAS-Ansatz und wurde publiziert [Sc19a].
- In Zusammenarbeit mit der Universität Prag wurde ein Konzept zur Semantifizierung von Lasttests erarbeitet. Basierend auf bestimmten Kontexten, die das Verhalten der Endnutzer beeinflussen, werden die fachlich und semantisch am besten zum Kontext passenden Verhalten automatisch selektiert. Dazu wurden verschiedene Clustering-basierte Ansätze des inkrementellen Lernens der Lastmodelle entwickelt und verglichen. Zudem wurde eine Beschreibungssprache für Kontexte entwickelt. Um basierend auf den erlernten Lastmodellen das erwartete Lastverhalten für einen vom Nutzer in dieser Sprache definierten Kontext zu bestimmen, werden die Zeitserienvorhersagewerkzeuge Telescope [Ba20] und Facebook Prophet [TL18] verwendet. Anschließend werden die relevanten Lastszenarien aus der Vorhersage extrahiert. Dadurch können Lasttests generiert werden, die präzise die im jeweiligen Kontext relevante Last testen und somit Ausführungszeit sparen, da im Kontext weniger relevante Lasten ausgelassen oder niedriger priorisiert werden können. Der Ansatz wurde auf einer einschlägigen Konferenz eingereicht und ist aktuell in Begutachtung [Sc20d].
- Im Rahmen der Integration mit BenchFlow (siehe AP 1) wurde ein Konzept des so genannten „behavior-driven" Lasttestens ausgearbeitet. Lasttests werden in natürlicher Sprache definiert, die einem Muster aus Vorbedingungen („given"), Änderungen der Vorbedingungen („when") und erwartetem Resultat („then") folgt. Der Ansatz wurde unter Mithilfe eines assoziierten Partners verfeinert und evaluiert. Es ist eine Publikation erfolgt [Sc19b].

3.1.3 AP 3: Kontinuierliches Performance-Regressionstesten

In diesem Arbeitspaket erfolgt die Integration der in AP 1 und AP 2 entwickelten Ansätze in die Infrastruktur der kontinuierlichen Softwareentwicklung.

Die wesentlichen Aktivitäten und Ergebnisse sind:

- Bereits zu Projektbeginn wurde mit der Entwicklung eines Software-Prototypen für das *ContinuITy*-Framework begonnen. Das Framework sollte als Integrations- und

Evaluationsplattform für die bestehenden sowie im Projekt entwickelten Ansätze dienen und die Abläufe automatisieren und in die Infrastruktur der kontinuierlichen Softwareentwicklung integrierbar machen. Es umfasst die meisten der entworfenen Ansätze und stellt eine webbasierte REST-Schnittstelle zur Verfügung, über die Lasttests generiert werden können. Dadurch können die Ansätze gut in Build-Pipelines wie z. B. Jenkins oder Testautomatisierungswerkzeuge wie BenchFlow integriert werden. Es können nun automatisiert Lasttests basierend auf einer Beschreibung des Kontexts sowie der zu testenden Services generiert werden. Durch das inkrementelle Lernen der Lastmodelle (siehe AP 2) konnte die Effizienz sowie der Grad der Automatisierung gesteigert werden. Insbesondere konnte so der komplexe Datensatz der Universität Prag analysiert und zu Evaluationszwecken verwendet werden. Das Framework ist als Open-Source-Software verfügbar.

Im Laufe des Projekts erfolgte eine enge Zusammenarbeit mit den Entwicklern des BenchFlow-Projekts. Der Hauptentwickler, Vincenzo Ferme (Doktorand an der Universität Lugano), war zwischenzeitlich im Kontext eines Forschungsaufenthalts im Projekt als Mitarbeiter tätig. Es wurde ein Konzept für eine Integration von ContinuITy und BenchFlow ausgearbeitet (siehe AP 1). Dadurch konnte die Integration in die kontinuierliche Softwareentwicklung verbessert werden.

- In einer gemeinsamen Arbeit mit dem assoziierten Partner Dr. Avritzer sowie Kollegen der Universitäten in Bozen-Bolzano (Italien), Lugano (Schweiz) und später auch Rio de Janeiro (Brasilien) wurde ein Ansatz zum quantitativen Vergleich von Konfigurationsalternativen von Microservices auf Basis von Lasttests und Monitoringdaten entwickelt sowie in einer Software namens PPTAM umgesetzt. Obwohl der ursprüngliche Fokus auf der Analyse von Performance und Skalierbarkeit lag, wurden Erweiterungen zur Analyse von Security-Attacken entwickelt. Die Ergebnisse wurden in Publikationen veröffentlicht [Av18, Av19, Av20]; die Software ist ebenfalls unter einer Open-Source-Lizenz verfügbar.

3.1.4 AP 4: Automatisierte Diagnose von Regressionen

Ziel des Arbeitspakets war die Diagnose, welche Änderung(en) zur erkannten Performance-Regression geführt haben, sowie die Berechnung und Aufbereitung der Regressionsauswirkungen auf die Performance in einer Form, dass sie von Entwicklern verstanden werden kann.

Die wesentlichen Aktivitäten und Ergebnisse sind:

- Es wurden Ansätze entwickelt, um die Verständlichkeit der Definition von Performance-(Regressions-)tests sowie der Verständlichkeit der Ergebnisse zu verbessern. Zur interaktiven textuellen Definition und Ausführung von Lasttests wurde eine Nutzerschnittstelle für Chat-Bots (PerformoBot) entwickelt und in einer Nutzerstudie

evaluiert. Diese Schnittstelle lässt sich in Team-Messengern (z. B. Slack) integrieren, welche verbreitete Werkzeuge der kontinuierlichen Softwareentwicklung sind. Diese Schnittstelle baut auf den ebenfalls im Kontext dieses Projekts entwickelten Vizard-Ansatz auf, der die Automatisierung der Erstellung der Lasttestskripte sowie die automatischen Generierung von Testberichten umsetzt. Diese Berichte erläutern Erklärungen als Kombination von natürlichsprachlichem Text und eingebetteten sowie verlinkten Diagrammen. Die Arbeiten wurden in Zusammenarbeit mit Kollegen der Universität Duisburg-Essen sowie Universität Würzburg durchgeführt und mündeten in entsprechende Publikationen (und Vorträge) auf einschlägigen Konferenzen [Ok19, Ok20].

- Es wurden mehrere Arbeiten zum Thema Performance-Regressionstesten für Microservices durchgeführt. Zunächst wurden in einem Laborexperiment existierende Performance-Regressionsansätze aus der Literatur auf deren Tauglichkeit in Microservice-Umgebungen untersucht. Innerhalb der SPEC Research Group (DevOps Performance Working Group) gehören hierzu eine Umfrage in der Industrie sowie umfangreiche Experimente zur Erkennung von Regressionen in Microservices. In diesem Kontext wurde mit Universitäten in Kanada (University of Alberta in Edmonton und Concordia University in Montréal) sowie den Universitäten in Kiel und Würzburg kooperiert. Gemeinsam mit Kollegen wurde eine Teilgruppe zum Performance-Testen für Microservices ins Leben gerufen, in der die genannten Arbeiten durchgeführt wurden. Die genannten Arbeiten führten zu Konferenzpublikationen [Be19, Ei20]. Zum Projektende wurde begonnen, diese Arbeiten auf Anwendungen zu erweitern, die nach dem Function-as-a-Service-Prinzip entworfen werden. Ferner erfolgte eine Zusammenarbeit (inkl. Forschungsaufenthalt) mit dem Gran Sasso Institute (GSSI) in L'Aquila zur Diagnose von Performance-Problemen durch dynamische Analyse.

3.1.5 AP 5: Validierung und Evaluation von ContinuITy

In diesem Arbeitspaket erfolgte die Bewertung der in den Arbeitspaketen AP 1–4 entwickelten Ansätze hinsichtlich qualitativer und quantitativer Fragestellungen. Die Evaluation erfolgte in Form von Labor- und Fallstudien.

Zum Zwecke der Evaluation wurden die entwickelten Ansätze prototypisch als Open-Source-Software implementiert. Der Quellcode [Sc20c, SD20, Sc20a] sowie die zur einfachen Ausführung bereitgestellten Docker-Images [Sc20b] sind online verfügbar. Ebenso wird in den jeweiligen Publikationen auf die öffentlich zugänglichen Evaluierungsdaten verwiesen.

Laborstudien

- Die in AP 2 entwickelte Erweiterung der WESSBAS-DSL um die Spezifikation von Eingabedaten wurde anhand der Broadleaf Heat Clinic evaluiert. Die Heat Clinic

ist eine Showcase-Anwendung der Broadleaf Commerce, welches ein Framework zur Erstellung von Web-Shop-Anwendungen ist. Es konnte gezeigt werden, dass der Aufwand, manuelle Änderungen an einem generierten Lasttest bei einem sich ändernden Umfeld vorzunehmen, signifikant verringert werden kann. Zudem wurde eine vollständige Automatisierung der Lasttestgenerierung erreicht, was den Einsatz in „Continuous Integration"- und „Continuous Delivery"-Pipelines erlaubt. Die Ergebnisse wurden in [SvHW20] publiziert.

- Die genannte Erweiterung der WESSBAS-DSL wurde in einer weiteren Studie mit der Nexus-Repository-Plattform der *Sonatype Inc.* durchgeführt. Neben über 100.000 Installationen in unterschiedlichsten Organisationen stellt Nexus die Basis für Maven Central dar — das von Sonatype betriebene und weltweit größte Repository für Open-Source-Komponenten. Die durchgeführten Experimente konnten die mit der Heat Clinic gewonnenen Erkenntnisse bekräftigen und bieten zusätzliche Evidenz, insbesondere im Bezug auf die Anwendung des Ansatzes in realistischen Szenarien. Die Ergebnisse wurden in [SvHW20] publiziert.
- Die Ansätze zur Microservice-basierten Modularisierung (siehe AP 2) wurden in einem Experiment mit der „Sock Shop Microservices Demo", die derzeit als eine der repräsentativsten Beispielanwendungen für Microservices gilt, evaluiert. Dazu wurden die leistungsstarken Server-Ressourcen im Future SOC Lab des Hasso-Plattner-Instituts (HPI) verwendet. In den untersuchten Szenarien konnten beide Ansätze den Ressourcenverbrauch deutlich verringern, ohne die erzeugte Last signifikant zu beeinflussen. Die Ergebnisse wurden in [Sc19a] publiziert.
- Die Semantifizierung von Lasttests (siehe AP 2) wurde anhand der aufgezeichneten Last des Studenteninformationssystems (SIS) der Charles-Universität Prag evaluiert. Der zur Evaluation verwendete Datensatz der aufgezeichneten Last des SIS wurde publiziert und ist öffentlich zugänglich [MT19]. Er wurde zum Zwecke der Evaluation der Semantifizierung analysiert und weiterverarbeitet. Insbesondere wurde das inkrementelle Lernen von Lastmodellen eingesetzt. Mittels dieser Lastmodelle wurde die Semantifizierung evaluiert. Die Ergebnisse wurden zur Publikation eingereicht und sind aktuell in Begutachtung [Sc20d].
- Im AP 3 wurden weitere umfangreiche Experimente mit der Sock-Shop-Anwendung durchgeführt. Die Experimente wurden im Cluster der Universität Stuttgart, in einem Cluster der Universität Bozen-Bolzano sowie im HPI durchgeführt. Die Ergebnisse wurden in [Av18, Av20] publiziert.
- Für den Ansatz zur Berichterstellung wurde eine erste Evaluation in Form eines kontrollierten Experiments mit Experten als Pilotstudie durchgeführt. Experten der Novatec haben als Probanden teilgenommen. Die Ergebnisse wurden in [Ok19] publiziert.
- Im Kontext der Aktivitäten mit der SPEC Research Group wurde eine weitere Laboranwendung, nämlich der TeaStore genutzt [Ei20].

Fallstudien und Umfragen

- In Zusammenarbeit mit einem assoziierten Partner wurde eine Fallstudie für den Ansatz des „behavior-driven" Lasttestens ausgearbeitet und durchgeführt. Der Partner hat Daten zur Verfügung gestellt, die durch das Konsortium analysiert werden und zur Evaluation der in AP 1 und AP 2 entwickelten Ansätze verwendet werden. Die entwickelte Beschreibungssprache konnte erfolgreich eingesetzt werden, und die Ergebnisse wurden bereits publiziert [Sc19b].
- Die in AP 2 entwickelte Erweiterung der WESSBAS-DSL um die Spezifikation von Eingabedaten wurde mit vier Kundenprojekten der Novatec evaluiert. Mithilfe eines Erweiterungsmechanismus konnten sämtliche Eingabedaten spezifiziert werden. Das untermauert die Einsatzfähigkeit des entwickelten Ansatzes in industriellen Szenarien. Die Ergebnisse wurden in [SvHW20] publiziert.
- Im Rahmen der Evaluierung der Semantifizierung von Lasttests wurde eine Fallstudie durchgeführt, die unter anderem Expertenbefragungen umfasste. Es konnte gezeigt werden, dass die entwickelte Beschreibungssprache für Kontexte ausreichend ausdrucksstark für realistische Szenarien ist. Ein solches Szenario war die COVID-19-Pandemie, die die Charles-Universität Prag zu Anpassungen der Lehre zwang, wie z. B. die Ersetzung der Präsenzlehre durch Online-Unterricht und damit verbundene verstärkte Nutzung des SIS. Die Ergebnisse sind aktuell in Begutachtung [Sc20d].
- Die Evaluation des Chat-Bot-basierten Nutzerschnittstelle erfolgte in Form einer (Online-)Nutzerstudie mit ca. 50 Teilnehmern [Ok20].

3.1.6 AP 6: Projektkoordination und Dissemination

Projektkoordination Dieser Teil des Arbeitspakets umfasst sämtliche Aktivitäten die mit der Organisation, Erfolgskontrolle, Steuerung und Berichterstattung (Zwischenberichte und Abschlussbericht) des Projektes verbunden sind.

Das Projekt wurde durch wöchentliche Treffen in Form von persönlichen Besprechungen, Telefon-/Videokonferenzen und Workshops sowie Abstimmungen über andere elektronische Kanäle (E-Mail, Chat, Atlassian Confluence etc.) geführt. Das Vorgehensmodell, das für die Projektabwicklung verwendet wurde, ist angelehnt an agile Softwareentwicklungsmethoden, wie beispielsweise Scrum, und durch eine iterative Herangehensweise charakterisiert. Ähnlich zur Idee von Daily-Scrum-Meetings wurden in diesen Treffen der aktuelle Arbeitsfortschritt aufgezeigt, Probleme diskutiert und die Arbeitsplanung abgestimmt. An diesen Treffen nahmen in der Regel alle am Projekt beteiligten Personen von der Novatec und der Universität Stuttgart teil. Die räumliche Nähe der Universität Stuttgart und der Novatec Consulting GmbH ermöglichte eine enge und sehr gute Kooperation der Projektpartner untereinander, wodurch auch die Durchführung von kurzfristig organisierten Treffen möglich und besonders hervorzuheben ist.

Im Confluence-System existiert eine umfassende Dokumentation von Projektartefakten wie z. B. Besprechungsprotokolle und Entwurfsdokumente. Zusätzlich wurde ein dedizierter geschützter Bereich in Confluence zur Zusammenarbeit mit externen Partnern genutzt, zu denen auch Studenten gehören, die Themen im *ContinuITy*-Kontext bearbeitet haben.

Die Qualitätssicherung wurde durch das agile Vorgehensmodell sowie gemeinsam durchgeführte Reviews und Evaluationen in besonderem Maße unterstützt. Der Rahmen für die Qualitätssicherung war dabei durch die gemeinsamen Workshops gegeben.

Es wurden zudem interne *ContinuITy*-Klausurtagungen durchgeführt, davon zwei im Schloss Dagstuhl, dem weltweit bekannten Begegnungszentrum für Informatik:

- 21.—23. November 2018, Event 18473, `https://www.dagstuhl.de/18473`
- 08.—10. Januar 2020, Event 20023, `https://www.dagstuhl.de/20023`

Die Workshops wurden genutzt, um den bisherigen Projektverlauf zu reflektieren, einzelne Themen im Detail zu diskutieren und Pläne für den weiteren Projektverlauf zu entwickeln.

Die assoziierten Partner wurden konkret in das *ContinuITy*-Projekt eingebunden. Neben dem Einholen von Feedbacks sowie der Vorbereitung und Durchführung der Evaluation gab es auch eine intensive inhaltliche Einbindung.

Für die Unterstützung der Projekttätigkeiten kamen für verschiedene Belange unterschiedliche Werkzeuge zum Einsatz. Zur Qualitätssicherung von Code-Artefakten wurde ein Continuous-Integration-System (Travis CI) verwendet. Code-Artefakte und Dokumente sowie Änderungen wurden mit Hilfe eines Code-und-Dokument-Repository (GitHub und GitLab) verwaltet. Für den asynchronen Austausch der Projektpartner und für die Wissensdokumentation kam ein Wiki (Atlassian Confluence) zum Einsatz sowie ein Issue/Tickettracking-System (Atlassian JIRA) für die Aufgabenplanung. Diese Systeme waren den Projektpartnern frei zugänglich, leichtgewichtig und bereits im industriellen Kontext für das Projektmanagement und zur Projektabwicklung etabliert. Die Werkzeuge wurden als Software-as-a-Service genutzt.

Dissemination Dieser Teil des Arbeitspakets umfasst sämtliche Aktivitäten, die dazu dienen, die in *ContinuITy* erarbeiteten Forschungsergebnisse während der Projektdurchführung einer breiteren Öffentlichkeit zur Verfügung zu stellen. Hierzu zählen beispielsweise *i)* die Einrichtung und Pflege einer *ContinuITy*-Webpräsenz, *ii)* die Veröffentlichung der Projektergebnisse auf Veranstaltungen wie Messen und Fachtagungen sowie in Fachzeitschriften, *iii)* die Veröffentlichung von entwickelter Software.

Für das Projekt wurde eine Webpräsenz unter `http://continuity-project.github.io/` eingerichtet. Für das Projekt wurde außerdem ein Zugang bei der Social-Media-Plattformen Twitter eingerichtet.

Der Projektstart wurde im SPEC RG Newsletter (vol. 2, issue 3, 2018) angekündigt.

Im Rahmen des Projekt wurden zahlreiche Publikationen in Zeitschriften, Konferenzen und Workshops erstellt. Zudem wurden zahlreiche Vorträge, Demos und Tutorials gehalten. Eine Vielzahl von Studierenden wurden in das Projekt eingebunden und haben zum Projekterfolg beigetragen. Eine detaillierte Auflistung der in diesem Absatz genannten Punkte ist Abschnitt 3.6 zu entnehmen.

Ferner war das Konsortium während des Projektzeitraums in verschiedenen Rollen (Funktionen, Sponsoring etc.) an der Organisation von wissenschaftlichen Veranstaltungen beteiligt:

- ACM/SPEC International Conference on Performance Engineering (‘18, ‘19, ’20), `http://icpe-conference.org/`
- International Workshop on Quality-Aware DevOps (’18, ’19, ’20), `http://qudos-workshop.org/`
- Symposium on Software Performance (‘17, ’18, ’19), `http://performance-symposium.org`

3.2 Wichtigste Positionen des zahlenmäßigen Nachweises

Die Kosten bzw. Ausgaben entfielen zum weitaus größten Teil auf die Position 0837 (Personalkosten) und zu einem sehr kleinen Teil auf die Positionen 0838 (Reisekosten) und 0843 (sonstige allgemeine Verwaltungsausgaben für eine Open-Access-Publikationsgebühr). Auf die weiteren Positionen des Verwendungsnachweises entfielen keine Kosten bzw. Ausgaben.

3.3 Notwendigkeit und Angemessenheit der geleisteten Arbeit

Die automatische, effiziente, und in die kontinuierliche Softwareentlicklung eingebundene Generierung, Selektion und Auswertung von Lattests stellen ein für die wirtschaftliche und wissenschaftliche Verwertung und den Anschluss herausragendes Alleinstellungsmerkmal im Kontext des Performance-Testens dar. Gleichzeitig sind die notwendigen Projektarbeiten für ein KMU eine Forschungsinvestition mit nicht unerheblichem Risiko. Das Projekt *ContinuITy* hat die Grundlage für die wirtschaftliche Umsetzung geschaffen und wesentliche wissenschaftliche Beiträge geleistet. Das Geschäftsfeld Performance-Testen der Novatec Consulting GmbH ist nachhaltig gestärkt und das Projektkonsortium plant weitere gemeinsame Aktivitäten. Alle durchgeführten Projektarbeiten und Ergebnisse waren dazu notwendig und angemessen.

3.4 Darstellung des voraussichtlichen Nutzens

Wie im vorangegangenen Abschnitt erläutert, konnten während der Projektlaufzeit die erstellten Artefakte in Laborstudien und Fallstudien erfolgreich evaluiert werden. Zudem gab es bei der Präsentationen des Ansatzes vor Vertretern der Industrie und der Wissenschaft stets großes Interesse. Somit kann ein großes Marktpotential unterstellt werden.

3.5 Fortschritte bei anderen Stellen

Während des Projekts hat sich der Markt der Open-Source-Monitoring-Software stark weiterentwickelt. Basierend auf Standards wie OpenTracing, OpenCensus und OpenTelemetry — welches die Zusammenführung der erstgenannten Standards ist — haben sich eine Vielzahl spezialisierter Werkzeuge etabliert. Mittels der Standards können diese Werkzeuge dann zu einer Gesamtlösung komponiert werden, wie die von der Novatec Consulting GmbH initiierte OpenAPM-Initiative[8] dokumentiert. Durch die in AP 1 vorgenommene Anpassung von OPEN.xtrace an OpenTracing können die im *ContinuITy*-Projekt erzielten Ergebnisse bereits gut in diesen Kontext integriert werden, was zu breiten Einsatzmöglichkeiten führt.

3.6 Erfolgte und geplante Veröffentlichungen

Im Zusammenhang mit dem *ContinuITy*-Projekt sind die im Folgenden aufgelisteten Publikationen, Vorträge und studentischen Arbeiten erfolgt. Bei den Publikationen ist zusätzlich eine Referenz auf das enthaltene Literaturverzeichnis enthalten.

3.6.1 Publikationen

Zeitschriftenartikel

- [Av20] Avritzer, Alberto; Ferme, Vincenzo; Janes, Andrea; Russo, Barbara; van Hoorn, André; Schulz, Henning; Menasché, Daniel; Rufino, Vilc: Scalability Assessment of Microservice Architecture Deployment Configurations: A Domain-based Approach Leveraging Operational Profiles and Load Tests. Journal of Systems and Software, 165:110564, 2020
- [SvHW20] Schulz, Henning; van Hoorn, André; Wert, Alexander: Reducing the maintenance effort for parameterization of representative load tests using annotations. Software Testing, Verification and Reliability, 30(1):e1712, 2020

[8] `https://openapm.io/`

- [Br20] Brataas, Gunnar; Hanssen, Geir Kjetil; Herbst, Nikolas; van Hoorn, André: Agile Scalability Engineering: The ScrumScale Method. IEEE Software, 2020. To appear
- [Tr18] Trubiani, Catia; Bran, Alexander; van Hoorn, André; Avritzer, Alberto; Knoche, Holger: Exploiting load testing and profiling for Performance Antipattern Detection. Information and Software Technology, 95:329 – 345, 2018

Workshop- und Konferenzpublikationen

- [Sc20d]: Schulz, Henning; van Hoorn, André; Tůma, Petr; Okanović, Dušan: Context-tailored Workload Model Extraction for Continuous Representative Load Testing. Under review, 2020
- [Ei20] Eismann, Simon; Bezemer, Cor-Paul; Shang, Weiyi; Okanović, Dušan; van Hoorn, André: Microservices: A Performance Tester's Dream or Nightmare? In: Proceedings of the ACM/SPEC International Conference on Performance Engineering. ICPE '20, S. 138–149, 2020
- [Ok20] Okanović, Dušan; Beck, Samuel; Merz, Lasse; Zorn, Christoph; Merino, Leonel; van Hoorn, André; Beck, Fabian: Can a Chatbot Support Software Engineers with Load Testing? Approach and Experiences. In: Proceedings of the ACM/SPEC International Conference on Performance Engineering. ICPE '20, S. 120–129, 2020
- [Sc19a] Schulz, Henning; Angerstein, Tobias; Okanović, Dušan; van Hoorn, André: Microservice-Tailored Generation of Session-Based Workload Models for Representative Load Testing. In: 2019 IEEE 27th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS). S. 323–335, 2019
- [Sc19b] Schulz, Henning; Okanović, Dušan; van Hoorn, André; Ferme, Vincenzo; Pautasso, Cesare: Behavior-Driven Load Testing Using Contextual Knowledge - Approach and Experiences. In: Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering. ICPE '19, S. 265–272, 2019
- [Ok19] Okanović, Dušan; van Hoorn, André; Zorn, Christoph; Beck, Fabian; Ferme, Vincenzo; Walter, Jürgen: Concern-Driven Reporting of Software Performance Analysis Results. In: Companion of the 2019 ACM/SPEC International Conference on Performance Engineering. ICPE '19, S. 1–4, 2019
- [Be19] Bezemer, Cor-Paul; Eismann, Simon; Ferme, Vincenzo; Grohmann, Johannes; Heinrich, Robert; Jamshidi, Pooyan; Shang, Weiyi; van Hoorn, André; Villavicencio, Monica; Walter, Jürgen; Willnecker, Felix: How is Performance Addressed in DevOps? In: Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering. ICPE '19, S. 45–50, 2019

- [Av19] Avritzer, Alberto; Menasché, Daniel; Rufino, Vilc; Russo, Barbara; Janes, Andrea; Ferme, Vincenzo; van Hoorn, André; Schulz, Henning: PPTAM: Production and Performance Testing Based Application Monitoring. In: Companion of the 2019 ACM/SPEC International Conference on Performance Engineering. ICPE '19, S. 39–40, 2019
- [Av18] Avritzer, Alberto; Ferme, Vincenzo; Janes, Andrea; Russo, Barbara; Schulz, Henning; van Hoorn, André: A Quantitative Approach for the Assessment of Microservice Architecture Deployment Alternatives by Automated Performance Testing. In: Software Architecture - 12th European Conference on Software Architecture. Springer International Publishing, S. 159–174, 2018
- [SAvH18] Schulz, Henning; Angerstein, Tobias; van Hoorn, André: Towards Automating Representative Load Testing in Continuous Software Engineering. In: Companion of the 2018 ACM/SPEC International Conference on Performance Engineering. ICPE '18, S. 123–126, 2018

Bücher und Kapitel in Büchern

- [vHS18] van Hoorn, André; Siegl, Stefan: Application Performance Management: Measuring and Optimizing the Digital Customer Experience. SIGS DATACOM GmbH, 2018. e-Book

Poster und Demonstrationen

- Alberto Avritzer, Daniel Sadoc Menasché, Vilc Rufino, Barbara Russo, Andrea Janes, Vincenzo Ferme, André van Hoorn, Henning Schulz, PPTAM: Production and Performance Testing Based Application Monitoring (Poster). 10th ACM/SPEC International Conference on Performance Engineering (ICPE) 2019.
- Henning Schulz, André van Hoorn, Dušan Okanović, Stefan Siegl, Christoph Heger, Alexander Wert, Tobias Angerstein, Alper Hidiroglu, Manuel Palenga, Christoph Zorn, Vincenzo Ferme, Alberto Avritzer, ContinuITy: Automated Load Testing in DevOps (Poster), 10th ACM/SPEC International Conference on Performance Engineering (ICPE) 2019. **Best Poster Award.**

3.6.2 Vorträge und Tutorials

Neben den Vorträgen für die oben aufgeführten Publikationen auf Workshops und Konferenzen haben Mitglieder des Konsortiums die folgenden Vorträge gehalten.

Konferenz- und Workshopvorträge (ohne entsprechende Publikation)

- Henning Schulz and André van Hoorn, Representative Load Testing in Continuous Software Engineering: Automation and Maintenance Support, Software Engineering (SE 20), Innsbruck, Austria (2020)
- Samuel Beck, Lasse Merz, Christoph Zorn, Fabian Beck, Leonel Merino, Dušan Okanović, and André van Hoorn, "PerformoBot, please help me!" — Chatbot-supported Performance Evaluation, Symposium on Software Performance 2020, Würzburg, Germany
- André van Hoorn, Test-based Scalability and Resilience Assessment of Microservice-based Software Systems, Keynote at International Workshop on Governing Adaptive and Unplanned Systems of Systems (GAUSS @ ISSRE 2019), Berlin Germany
- André van Hoorn, Automated Load Testing in Continuous Software Engineering, ScrumScale Meeting, Oslo, Norway (2018)
- Vincenzo Ferme, Continuous Performance Testing for Microservices, HPI Future SOC Lab Day - Spring 2018, Potsdam, Germany
- Henning Schulz, André van Hoorn, Christoph Heger, and Alexander Wert, ContinuITy: Automated Performance Testing in Continuous Software Engineering, Symposium on Software Performance 2017, Karlsruhe, Germany

Tutorials

- Alberto Avritzer, André van Hoorn, Vincenzo Ferme, Henning Schulz, Barbara Russo, and Andrea Janes. Exploiting Operational Profile Data for Continuous Dependability Assessment in DevOps, dreistündiges Tutorial auf dem 29th IEEE International Symposium on Software Reliability Engineering (ISSRE 2018) in Memphis, TN, USA.

3.6.3 Studentische Arbeiten

Die im Folgenden aufgeführten Arbeiten wurden in der Regel gemeinsam von der Universität Stuttgart und der Novatec betreut.

Masterarbeiten

- Niko Stadelmaier. Using Software-Performance-Antipatterns and Profiling Traces to Perform Code Refactorings, University of Stuttgart, 2020.

- Alper Hidiroglu, Context-Aware Load Testing in Continuous Software Engineering, University of Stuttgart, 2019.
- Manuel Palenga, Declarative User Experience Regression Analysis in Continuous Performance Engineering, University of Stuttgart, 2018.
- Tobias Angerstein, Modularization of Representative Load Tests for Microservice Applications, University of Stuttgart, 2018.

Masterprojekte

- Samuel Beck, Lasse Merz, Christoph Zorn, Using Chatbots to Facilitate Performance Evaluations, Guided Research Project, University of Stuttgart, 2019

Bachelorarbeiten

- Christoph Zorn, Concern-driven Reporting of Declarative Performance Analysis Results Using Natural Language and Visualization, University of Stuttgart, 2018.

Software

- ContinuITy: https://github.com/ContinuITy-Project/ContinuITy
- Clustinator: https://github.com/ContinuITy-Project/clustinator
- Forecastic: https://github.com/ContinuITy-Project/forecastic
- PPTAM: https://github.com/pptam
- Docker-Images auf DockerHub: https://hub.docker.com/u/continuityproject

Literatur

[ABV16] Alcocer, Juan Pablo Sandoval; Bergel, Alexandre; Valente, Marco Tulio: Learning from Source Code History to Identify Performance Failures. In: Proceedings of the 7th ACM/SPEC International Conference on Performance Engineering (ICPE 2016). S. 37–48, 2016.

[Av18] Avritzer, Alberto; Ferme, Vincenzo; Janes, Andrea; Russo, Barbara; Schulz, Henning; van Hoorn, André: A Quantitative Approach for the Assessment of Microservice Architecture Deployment Alternatives by Automated Performance Testing. In: Software Architecture - 12th European Conference on Software Architecture. Springer International Publishing, S. 159–174, 2018.

[Av19] Avritzer, Alberto; Menasché, Daniel; Rufino, Vilc; Russo, Barbara; Janes, Andrea; Ferme, Vincenzo; van Hoorn, André; Schulz, Henning: PPTAM: Production and Performance Testing Based Application Monitoring. In: Companion of the 2019 ACM/SPEC International Conference on Performance Engineering. ICPE '19, S. 39–40, 2019.

[Av20] Avritzer, Alberto; Ferme, Vincenzo; Janes, Andrea; Russo, Barbara; van Hoorn, André; Schulz, Henning; Menasché, Daniel; Rufino, Vilc: Scalability Assessment of Microservice Architecture Deployment Configurations: A Domain-based Approach Leveraging Operational Profiles and Load Tests. Journal of Systems and Software, 165:110564, 2020.

[Ba04] Balsamo, Simonetta; Marco, Antinisca Di; Inverardi, Paola; Simeoni, Marta: Model-Based Performance Prediction in Software Development: A Survey. IEEE Trans. Software Eng., 30(5):295–310, 2004.

[Ba20] Bauer, André; Züfle, Marwin; Herbst, Nikolas; Kounev, Samuel; Curtef, Valentin: Telescope: An Automatic Feature Extraction and Transformation Approach for Time Series Forecasting on a Level-Playing Field. In: Proceedings of the 36th International Conference on Data Engineering (ICDE 2020). 2020.

[Be14] Bezemer, Cor-Paul; Milon, Elric; Zaidman, Andy; Pouwelse, Johan: Detecting and analyzing I/O performance regressions. Journal of Software: Evolution and Process, 26(12):1193–1212, 2014.

[Be15] Bergmann, Volker et al.: , ContiPerf. `https://github.com/lucaspouzac/contiperf`, 2015.

[Be19] Bezemer, Cor-Paul; Eismann, Simon; Ferme, Vincenzo; Grohmann, Johannes; Heinrich, Robert; Jamshidi, Pooyan; Shang, Weiyi; van Hoorn, André; Villavicencio, Monica; Walter, Jürgen; Willnecker, Felix: How is Performance Addressed in DevOps? In: Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering. ICPE '19, S. 45–50, 2019.

[Bo14] Bosch, Jan, Hrsg. Continuous Software Engineering. Springer, 2014.

[Br15] Brunnert, Andreas; van Hoorn, Andre; Willnecker, Felix; Danciu, Alexandru; Hasselbring, Wilhelm; Heger, Christoph; Herbst, Nikolas; Jamshidi, Pooyan; Jung, Reiner; von Kistowski, Joakim; Koziolek, Anne; Kroß, Johannes; Spinner, Simon; Vögele, Christian; Walter, Jürgen; Wert, Alexander: Performance-oriented DevOps: A Research Agenda. Bericht SPEC-RG-2015-01, SPEC Research Group — DevOps Performance Working Group, Standard Performance Evaluation Corporation (SPEC), 2015.

[Br20] Brataas, Gunnar; Hanssen, Geir Kjetil; Herbst, Nikolas; van Hoorn, André: Agile Scalability Engineering: The ScrumScale Method. IEEE Software, 2020. To appear.

[BS11] Bouillet, Patrice; Siegl, Stefan: inspectIT: Java-Performance auf dem Prüfstand. Java Magazin, 2011.

[BWZ15] Bass, Len; Weber, Ingo; Zhu, Liming: DevOps: A Software Architect's Perspective. Addison-Wesley Prof., 2015.

[Cl10] Clark, Mike: , JUnitPerf. `https://github.com/clarkware/junitperf`, 2010.

[CMT16] Calzarossa, Maria Carla; Massari, Luisa; Tessera, Daniele: Workload Characterization: A Survey Revisited. ACM Comput. Surv., 48(3):48, 2016.

[Ei20] Eismann, Simon; Bezemer, Cor-Paul; Shang, Weiyi; Okanović, Dušan; van Hoorn, André: Microservices: A Performance Tester's Dream or Nightmare? In: Proceedings of the ACM/SPEC International Conference on Performance Engineering. ICPE '20, S. 138–149, 2020.

[FP18] Ferme, Vincenzo; Pautasso, Cesare: A Declarative Approach for Performance Tests Execution in Continuous Software Development Environments. In: Proceedings of 8th ACM/SPEC International Conference on Performance Engineering (ICPE 2018). ACM, 2018.

[GG15] Grove, Andy; Gardner, Brent: , JPerf. `https://github.com/AgilData/jperf`, 2015.

[HCS15] Haight, Cameron; Cappelli, Will; Silva, Federico De: , Magic Quadrant for Application Performance Monitoring Suites, 2015.

[He16] Heger, Christoph; van Hoorn, André; Okanović, Dušan; Siegl, Stefan; Wert, Alexander: Expert-Guided Automatic Diagnosis of Performance Problems in Enterprise Applications. In: Proceedings of the 12th European Dependable Computing Conference (EDCC '16). IEEE, 2016.

[He18] Heger, Christoph; van Hoorn, André; Okanović, Dušan; Siegl, Stefan; Vögele, Christian; Wert, Alexander: , diagnoseIT: Expertengestützte automatische Diagnose von Performance-Problemen in Enterprise-Anwendungen (Abschlussbericht), Januar 2018.

[HHF13] Heger, Christoph; Happe, Jens; Farahbod, Roozbeh: Automated Root Cause Isolation of Performance Regressions During Software Development. In: Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering. ICPE '13. ACM, S. 27–38, 2013.

[Ho09] van Hoorn, André; Rohr, Matthias; Hasselbring, Wilhelm; Waller, Jan; Ehlers, Jens; Frey, Sören; Kieselhorst, Dennis: Continuous Monitoring of Software Services: Design and Application of the Kieker Framework. Bericht TR-0921, Department of Computer Science, University of Kiel, Germany, 2009.

[Ho13] Horký, Vojtěch; Haas, František; Kotrč, Jaroslav; Lacina, Martin; Tůma, Petr: Performance Regression Unit Testing: A Case Study. In: Computer Performance Engineering: 10th European Workshop, EPEW 2013, Venice, Italy, September 16-17, 2013. Proceedings. Springer Berlin Heidelberg, S. 149–163, 2013.

[JH15] Jiang, Zhen Ming; Hassan, Ahmed E.: A Survey on Load Testing of Large-Scale Software Systems. IEEE Trans. Software Eng., 41(11):1091–1118, 2015.

[Ko10] Koziolek, Heiko: Performance evaluation of component-based software systems: A survey. Perform. Eval., 67(8):634–658, 2010.

[Li09] Liggesmeyer, Peter: Software-Qualität - Testen, Analysieren und Verifizieren von Software (2. Aufl.). Spektrum Akademischer Verlag, 2009.

[MT19] Maňásek, Martin; Tůma, Petr: , Charles University SIS Access Log Dataset, 2019. Zenodo. `http://doi.org/10.5281/zenodo.3241445`.

[Ne15] Newman, Sam: Building Microservices—Designing Fine-Grained Systems. O'Reilly Media, 2015.

[Ok16] Okanović, Dušan; van Hoorn, André; Heger, Christoph; Wert, Alexander; Siegl, Stefan: Towards Performance Tooling Interoperability: An Open Format for Representing Execution Traces. In: Proceedings of the 13th European Workshop on Performance Engineering (EPEW '16). LNCS. Springer, 2016.

[Ok19] Okanović, Dušan; van Hoorn, André; Zorn, Christoph; Beck, Fabian; Ferme, Vincenzo; Walter, Jürgen: Concern-Driven Reporting of Software Performance Analysis Results. In: Companion of the 2019 ACM/SPEC International Conference on Performance Engineering. ICPE '19, S. 1–4, 2019.

[Ok20] Okanović, Dušan; Beck, Samuel; Merz, Lasse; Zorn, Christoph; Merino, Leonel; van Hoorn, André; Beck, Fabian: Can a Chatbot Support Software Engineers with Load Testing? Approach and Experiences. In: Proceedings of the ACM/SPEC International Conference on Performance Engineering. ICPE '20, S. 120–129, 2020.

[Pr16] Prokopec, Aleksandar et al.: , ScalaMeter. `https://github.com/scalameter/scalameter`, 2016.

[RK16] Reichelt, David Georg; Kühne, Stefan: Empirical Analysis of Performance Problems on Code Level. In: Proceedings of the 7th ACM/SPEC on International Conference on Performance Engineering (ICPE 2016). ACM, S. 117–120, 2016.

[SAvH18] Schulz, Henning; Angerstein, Tobias; van Hoorn, André: Towards Automating Representative Load Testing in Continuous Software Engineering. In: Companion of the 2018 ACM/SPEC International Conference on Performance Engineering. ICPE '18, S. 123–126, 2018.

[Sc19a] Schulz, Henning; Angerstein, Tobias; Okanović, Dušan; van Hoorn, André: Microservice-Tailored Generation of Session-Based Workload Models for Representative Load Testing. In: 2019 IEEE 27th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS). S. 323–335, 2019.

[Sc19b] Schulz, Henning; Okanović, Dušan; van Hoorn, André; Ferme, Vincenzo; Pautasso, Cesare: Behavior-Driven Load Testing Using Contextual Knowledge - Approach and Experiences. In: Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering. ICPE '19, S. 265–272, 2019.

[Sc20a] Schulz, Henning: , ContinuITy-Project/Forecastic: Release at Project End (v0.5.3), 2020. Zenodo. `https://doi.org/10.5281/zenodo.3966834`.

[Sc20b] Schulz, Henning: , Docker Images for ContinuITy, 2020. Zenodo. `https://doi.org/10.5281/zenodo.3966908`.

[Sc20c] Schulz, Henning; Angerstein, Tobias; Palenga, Manuel; Hidiroglu, Alper: , ContinuITy-Project/ ContinuITy: Release at Project End (v2.9.346), 2020. Zenodo. `https://doi.org/10.5281/zenodo.3966805`.

[Sc20d] Schulz, Henning; van Hoorn, André; Tůma, Petr; Okanović, Dušan: Context-tailored Workload Model Extraction for Continuous Representative Load Testing. Under review, 2020.

[SD20] Schulz, Henning; Dang, An: , ContinuITy-Project/Clustinator: Release at Project End (v0.7.4), 2020. Zenodo. `https://doi.org/10.5281/zenodo.3966829`.

[SvHW20] Schulz, Henning; van Hoorn, André; Wert, Alexander: Reducing the maintenance effort for parameterization of representative load tests using annotations. Software Testing, Verification and Reliability, 30(1):e1712, 2020.

[TL18] Taylor, Sean J.; Letham, Benjamin: Forecasting at Scale. 72(1):37–45, 2018.

[Tr18] Trubiani, Catia; Bran, Alexander; van Hoorn, André; Avritzer, Alberto; Knoche, Holger: Exploiting load testing and profiling for Performance Antipattern Detection. Information and Software Technology, 95:329 – 345, 2018.

[vHS18] van Hoorn, André; Siegl, Stefan: Application Performance Management: Measuring and Optimizing the Digital Customer Experience. SIGS DATACOM GmbH, 2018. e-Book.

[vHWH12] van Hoorn, André; Waller, Jan; Hasselbring, Wilhelm: Kieker: A Framework for Application Performance Monitoring and Dynamic Software Analysis. In: Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering (ICPE '12). ACM, 2012.

[Vö16] Vögele, Christian; van Hoorn, André; Schulz, Eike; Hasselbring, Wilhelm; Krcmar, Helmut: WESSBAS: Extraction of Probabilistic Workload Specifications for Load Testing and Performance Prediction—A Model-Driven Approach for Session-Based Application Systems. Journal on Software and System Modeling (SoSyM), 2016.

[WFP07] Woodside, C. Murray; Franks, Greg; Petriu, Dorina C.: The Future of Software Performance Engineering. In: International Conference on Software Engineering, (ICSE 2007), Workshop on the Future of Software Engineering (FOSE 2007). S. 171–187, 2007.