



Fakultät Wirtschaft

Studiengang Wirtschaftsinformatik Software Engineering

**Integrations- und
Bereitstellungsautomatisierung von
Cloud-Anwendungen für
Composable-Enterprise-Architekturen im
Kontext der SAP Business Technology Platform**

Bachelorarbeit

Im Rahmen der Prüfung zum Bachelor of Science (B. Sc.)

Verfasser:	Rafael Martin
Kurs:	WI SE-B 2020
Dualer Partner:	SAP SE, Walldorf
Betreuer der Ausbildungsfirma:	Klaus Räwer
Wissenschaftlicher Betreuer:	Herr Ulrich Wolf
Abgabedatum:	08.05.2023

Selbstständigkeitserklärung

Ich versichere hiermit, dass ich die vorliegende Bachelorarbeit mit dem Thema:

**Integrations- und Bereitstellungsautomatisierung von
Cloud-Anwendungen für Composable-Enterprise-Architekturen im
Kontext der SAP Business Technology Platform**

selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Mannheim, 08.05.2023, _____

Rafael Martin

Inhaltsverzeichnis

Selbstständigkeitserklärung	II
Abkürzungsverzeichnis	V
Abbildungsverzeichnis	VII
Tabellenverzeichnis	VIII
1 Einleitung	1
1.1 Motivation und Problemstellung	1
1.2 Zielsetzung und Abgrenzung	2
1.3 Aufbau der Arbeit	3
2 Grundlagen und Begriffserklärungen	5
2.1 Die Composable-Enterprise-Architektur	5
2.1.1 Begriffserklärung und Abgrenzung	5
2.1.2 Technologische Konzepte des Composable-Enterprises	8
2.2 Integration und Bereitstellung einer Cloud-Anwendung	10
2.2.1 Agile und DevOps als moderne Softwareentwicklungskonzepte	10
2.2.2 Automatisierung der Integrations- und Bereitstellungsprozesse	12
2.2.3 Strategien zur Bereitstellung von Neuentwicklungen	18
3 Methodische Vorgehensweise	20
3.1 Semistrukturierte Leitfadeninterviews zur Erhebung qualitativer Daten	20
3.2 Evaluation von Integrations- und Bereitstellungs-Tools unter Anwen- dung des Analytischen Hierarchieprozesses	22
4 Evaluation der Integrations- und Bereitstellungs-Pipelines unter Anwendung des Analytischen Hierarchieprozesses	26
4.1 Definition der Entscheidungskriterien	26
4.2 Festlegung der Bewertungsmetriken	33
4.3 Ermittlung der Gewichtungsfaktoren	34

4.4	Bewertung der Entscheidungsalternativen	36
5	Entwicklung einer ganzheitlichen Bereitstellungsstrategie	51
6	Schlussbetrachtung	52
6.1	Fazit und kritische Reflexion	52
6.2	Ausblick auf zukünftige Entwicklungen	52
	Literaturverzeichnis	IX
	Anhang	XIII

Abkürzungsverzeichnis

XP	Extreme Programming
DevOps	Development & Operations
CI/CD	Continuous Integration and Continuous Delivery
CI	Continuous Integration
CD	Continuous Delivery
DoD	Definition of Done
E2E-Tests	End-to-End-Tests
PBC	Packaged-Business-Capability
CEA	Composable-Enterprise-Architektur
MACH	Microservices, APIs, Cloud-native, Headless
EDA	Event-driven Architecture
NIST	National Institute of Standards and Technology
SaaS	Software-as-a-Service
PaaS	Platform-as-a-Service
IaaS	Infrastructure-as-a-Service
SAP CI/CD	SAP Continuous Integration and Delivery
AHP	Analytischer Hierarchieprozesses
SAP BTP	SAP Business Technology Plattform
MTA	Multi-Target Application
SAP CTM	SAP Cloud Transport Management
SAP BAS	SAP Business Application Studio
DAST	Dynamic Application Security Testing
SAST	Static Application Security Testing
ERP	Enterprise-Ressourcen-Planning
CRM	Customer-Relationship-Management
TCO	Total Cost of Owner Ship
SSO	Single-Sign-On
CIO	Chief Information Officer

SAP DTS SAP Data Technology Services

Abbildungsverzeichnis

1	Aufbau der Arbeit	4
2	Entstehung einer Composable-Enterprise-Architektur	5
3	Bestandteile eines Composable-ERP-Systems	6
4	Technische Realisierung der Composable-Enterprise-Architektur . . .	8
5	Exemplarische Abfolge eines agilen Entwicklungszykluses	10
6	Aktivitäten im CI/CD-Prozess	12
7	Integration der CI/CD-Pipeline mit dem Versionskontrollsystem . . .	13
8	Hierarchische Darstellung von Softwaretests	14
9	Strategien zur Bereitstellung von Software	18
10	Exemplarische Darstellung der hierarchischen Entscheidungsstruktur im AHP	22
11	Exemplarische Darstellung der Paarvergleichsmatrix im AHP	23
12	AHP-Entscheidungsstruktur zur Bewertung von CI/CD-Pipelines . .	27
13	Qualitative Bewertungsmetrik für AHP	33
14	Quantitative Bewertungsmetrik für AHP	34
15	Gewichtungsfaktoren der AHP-Entscheidungskriterien	35
16	SAP Cloud Transportmanagement	39
17	Pipeline-Monitoring mit Kibana und Jenkins	40
18	CEA-Szenario für Perfomance-Tests	44
19	Modularer Aufbau einer CI/CD-Pipeline	46
20	Zeitliche Darstellung der Herbeiführung von Kundennutzen bei der Entwicklung von IT-Services	XV

Tabellenverzeichnis

1	Integration- und Delivery-Zeit in Sekunden	45
2	Ergebnistabelle zum AHP	50

1 Einleitung

1.1 Motivation und Problemstellung

In der heutigen Zeit erweist sich Flexibilität als eine immer bedeutendere Determinante für den Erfolg von Unternehmen. So müssen diese stets in der Lage sein, sich effizient an verändernde Marktbedingungen anzupassen. Unternehmen sehen sich deshalb vornehmlich damit konfrontiert, die in Enterprise-Ressource-Planning-Systemen (ERP-Systemen) abgebildeten Prozesse auf externe Einflüsse, wie Kundenbedürfnisse, rechtliche Vorschriften und neue Technologien auszurichten. Mit der *Composable-Enterprise-Architektur (CEA)* hat sich in den vergangenen Jahren ein Konzept etabliert, welches Unternehmen bei der Bewältigung dieser Disruptionen unterstützen soll. In einer CEA werden modulare Software-Komponenten zu einem homogenen Gesamtsystem konsolidiert [0]. Martin Henning, Head of New Ventures and Technologies bei der SAP, bemerkt, dass Unternehmen mit diesem Architekturkonzept nicht länger auf „rigide Systeme“ angewiesen, sondern vielmehr in der Lage sind, Geschäftsprozesse auf Grundlage einzelner Software-Bausteine zusammenzusetzen [0]. Bei einer Änderung der Geschäftsanforderungen besteht somit die Möglichkeit einzelne Software-Komponenten dynamisch auszutauschen ohne, dass Anpassungen am Gesamtsystem erforderlich sind. Dieser Composable-Trend ist nicht nur bei der SAP, sondern ebenfalls bei anderen Unternehmen erkennbar. So wurde in Gartners Top-Trend-Forschung 2022 prognostiziert, dass bis zum Jahr 2024 80 Prozent der befragten Chief Information Officers (CIOs) die modulare Gestaltung von Geschäftsprozessen als eine der fünf wichtigsten Gründe für betriebliches Wachstum betrachten werden [0]. Um die Geschäftsprozesse in einer CEA noch individueller auf die eigenen Bedürfnisse zuschneiden zu können, neigen Unternehmen dazu, die bestehende Architektur um eigenentwickelte modulare Bausteine zu erweitern. Damit Effizienz und Anpassungsfähigkeit vollständig ausgeschöpft werden kann, ist es unerlässlich, dass diese Bausteine schnell bereitgestellt und in das bestehende System integriert werden. Abhilfe schaffen soll dabei die in der Literatur als *Continuous Integration and Continuous Delivery (CI/CD)* bekannte Entwicklerpraktik. Damit

soll sichergestellt werden, dass Änderungen am Code häufiger und zuverlässiger in den Produktionssystemen bereitgestellt werden. Dies kann mithilfe einer CI/CD-Pipeline realisiert werden. Mit dieser ist es möglich, die in einer Entwicklungsabteilung anfallenden Software-Bereitstellungsprozesse vollständig zu automatisieren. Der *State of DevOps Report* zeigt, dass die in Bereitstellungssystemen abgewickelten Tests sowie das zyklische Ausliefern von Software zu einem signifikanten Rückgang der Ausfallrate in Produktionssystemen geführt hat [0]. Da CI/CD-Pipelines eine kontinuierliche Bereitstellung und Integration modularer Software-Komponenten ermöglichen, sind diese insbesondere für CEAs von großer Bedeutung. Um den spezifischen in einer CEA vorliegenden Bedürfnissen gerecht zu werden, benötigt die Implementierung von CI/CD-Prozessen jedoch im Vergleich zu herkömmlichen System-Architekturen eine differenzierte Herangehensweise. Damit die Unabhängigkeit einzelner Software-Komponenten gewährleistet werden kann, besteht für CEA die Notwendigkeit einer granularen und dezentralen Pipeline-Struktur. Dabei bleibt zu hinterfragen, ob gegenwärtige CI/CD-Tools in der Lage sind, diesen Anforderungen zu genügen.

1.2 Zielsetzung und Abgrenzung

Die technische Beratungsabteilung SAP Data Technology Service (SAP DTS) unterstützt Kunden bei der Implementierung einer CEA auf der SAP-Cloud-Plattform. Ein essenzielles Thema stellt in diesem Kontext ebenfalls die Beratung bei der Implementierung von Software-Bereitstellungsprozessen dar. Um diese Vorgänge zu automatisieren, werden von dem SAP DTS i.d.R. drei verschiedene CI/CD-Pipeline-Tools empfohlen. Dazu gehören Azure Pipelines, Jenkins und (SAP CI/CD). Ziel der Arbeit ist deshalb, zu evaluieren, welches dieser Tools den größten Mehrwert zur Bereitstellung von Cloud-Software für eine CEA liefert. Dafür soll ein Entscheidungs-Framework erstellt, anhand welchem die Lösungen bewertet werden. Es besteht die Möglichkeit, dass das mit dem Entscheidungs-Framework erhaltene Resultat nicht auf alle Projektkontexte übertragbar ist. Aus diesem Grund ist in der vorliegenden Arbeit ebenfalls vorgesehen, das Ergebnis in einer abschließenden Handlungsemp-

fehlung einzuordnen und abzugrenzen. Daraus resultiert folgende Forschungsfrage:
Welches Tool bietet zur Automatisierung der Bereitstellungsprozesse für Composable-Enterprise-Architekturen den größten Mehrwert?

Im Rahmen der Zielsetzung werden folgende Abgrenzungen vorgenommen: Auf der SAP-Cloud-Plattform können Anwendungen auf verschiedenen Laufzeitumgebungen betrieben werden. Dafür wird neben Neo und Kyma ebenfalls Cloud Foundry bereitgestellt. In der vorliegenden Arbeit wird dabei jedoch ausschließlich die Bereitstellung von Software in der Cloud-Foundry-Laufzeitumgebung untersucht. Zudem beschränkt sich die Analyse der CI/CD-Tools auf Anwendungen, welche auf den Programmier-Frameworks SAP CAP Node sowie SAP UI5 basieren. Diese Abgrenzung wird gezogen, da das SAP DTS ausschließlich in diesen Technologien berät.

1.3 Aufbau der Arbeit

Die theoretischen Grundlagen beginnen mit der Begriffsdefinition und Abgrenzung der CEA. Im Anschluss werden technologische Konzepte der CEA erläutert. Dabei wird dargelegt, wie CEs ihre Architektur gestalten müssen, um betriebswirtschaftliche Abläufe adäquat in der Unternehmenssoftware abzubilden. Im nachfolgenden Abschnitt werden im Rahmen der Softwareentwicklung anfallende Integrations- und Bereitstellungsprozesse beschrieben. Hierbei wird zunächst erläutert, wie die Entwicklungskonzepte *Agile* und *DevOps* zur Optimierung dieser Prozesse beitragen. Daraufhin wird dargestellt, wie CI/CD-Tools zur Automatisierung dieser Bereitstellungsabläufe eingesetzt werden können. Im Methodikteil wird das gewählte Vorgehen zu den Experteninterviews, welche zur Erhebung qualitativer Daten durchgeführt werden, erläutert. Des Weiteren wird der *Analytische Hierarchieprozesses (AHP)*, das Instrument zur Bestimmung des optimalen CI/CD-Tools beschrieben. Im Teil der Durchführung erfolgt die Anwendung der Methodik auf die in den Experteninterviews erhobenen Daten. So werden im Rahmen des AHP-Verfahrens Entscheidungskriterien festgelegt, welche im Anschluss gewichtet und zur Bestimmung eines optimalen CI/CD-Pipeline verwendet werden. In der folgenden Handlungsempfehlung wird eine ganzheitliche Bereitstellungsstrategie für CEA entwickelt. Dabei wird

das Ergebnis des AHP-Verfahrens analysiert und in Abhängigkeit verschiedener Unternehmensstrategien abgegrenzt. Abgerundet wird die Arbeit durch die Zusammenfassung der Erkenntnisse, einer kritischen Beleuchtung des Vorgehens und der Darstellung zukünftiger Ereignisse.

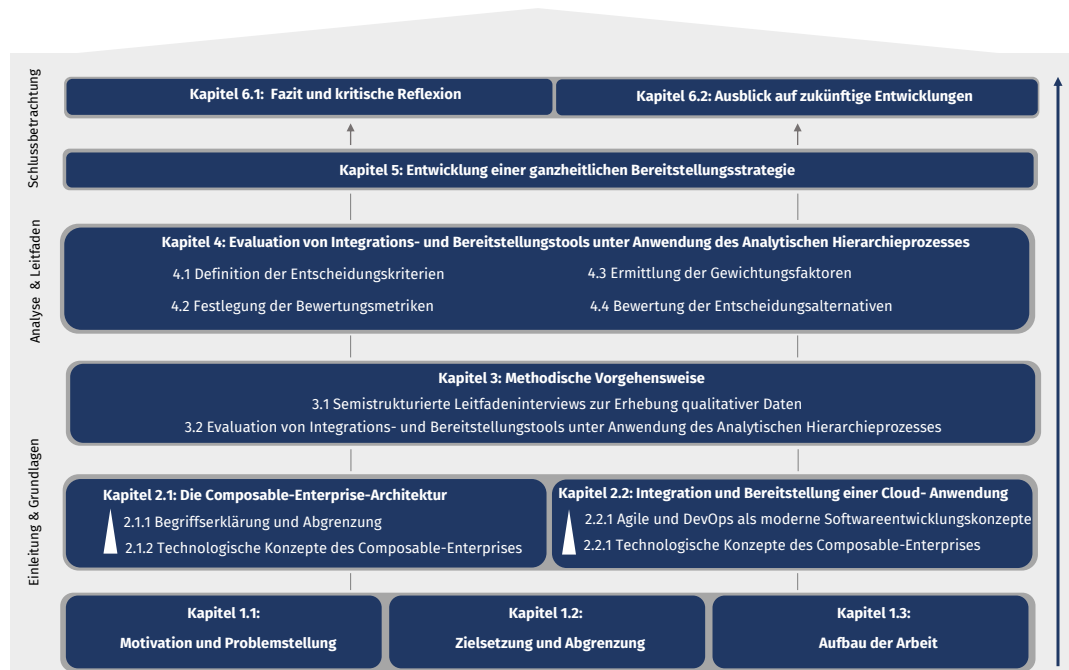


Abbildung 1: Aufbau der Arbeit. Eigene Darstellung.

2 Grundlagen und Begriffserklärungen

2.1 Die Composable-Enterprise-Architektur

2.1.1 Begriffserklärung und Abgrenzung

Flexibilität, Resilienz und Agilität. Nach Ansicht von Steve Denning, Managementberater und Autor der Forbes, stellen diese Eigenschaften wesentliche Faktoren dar, welcher „zur Steigerung der Wettbewerbsfähigkeit von Unternehmen beitragen“ [0]. Für Analystenhäuser wie Gartner stet dabei fest, dass es technologischer Innovation benötigt, um einhergehende Herausforderungen erfolgreich zu bewältigen und eine kontinuierliche Unternehmenstransformation voranzutreiben. Gartner empfiehlt dabei monolithische und starre Unternehmensarchitekturen, durch einen modularen Organisationsaufbau zu ersetzen. In seinen Veröffentlichungen verwendet Gartner für dieses Konzept den Begriff der **Composable-Enterprise-Architektur (CEA)**.

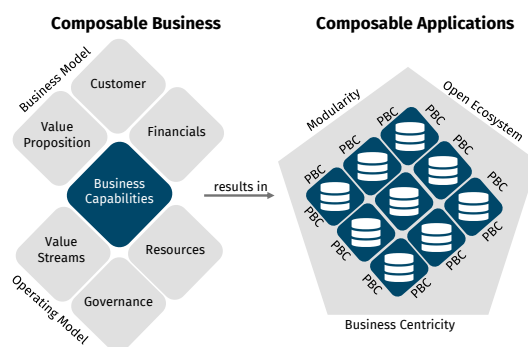


Abbildung 2: Entstehung einer Composable-Enterprise-Architektur. In Anlehnung an Schönstein [0].

In der Literatur wird die CEA dabei wie folgt definiert:

„The Composable-Enterprise-Architecture is an architecture that delivers business outcomes and adapts to the pace of business change. It does this through the assembly and combination of packaged business capabilities [0].“

Ein CE ist somit ein aus mehreren Bausteinen, sog. *Packaged-Business-Capability (PBC)* bestehendes Unternehmen. PBCs sind vorgefertigte Softwareelemente, welche jeweils eine bestimmte Geschäftsfunktion abdecken (s. Abb. 2). Das Konzept

der PBCs fußt dabei auf den drei Prinzipien der CEA: *Modulare Architektur*, *Offenes Ökosystem* und *Businesszentriertheit* [0]. Laut Gartner müssen Unternehmen nicht nur „akzeptieren, dass der disruptive Wandel zur Normalität gehört“. Vielmehr sollten diese den disruptiven Wandel als „Chance begreifen und ihn nutzen, um eine *modulare Architektur* zu implementieren“ [0]. In diesem Kontext wird von dem Analystenhaus Gartner ebenfalls der Begriff des *Composable-Enterprise-Ressourcen-Planning-Systems (Composable-ERP-Systems)* verwendet. Hierbei stellen die modularen Komponenten (PBCs) vorgefertigte Geschäftsfunktionen- bzw. prozesse dar, welche als Module in ein Composable-ERP-System integriert werden können. Diese PBCs können etwa Funktionen wie Finanzbuchhaltung, Einkauf, Verkauf, Lagerverwaltung, Produktion oder Personalmanagement enthalten. Ergibt sich eine Änderung in den Geschäftsanforderungen, ermöglicht diese Architektur ein flexibles und isoliertes Austauschen, Verändern sowie Weiterentwickeln einzelner PBCs [4, S. 315]. Durch die unabhängige Bereitstellung der Software-Komponenten ist es weiterhin möglich, einzelne Module einer CEA skalierbar zu gestalten ohne dabei die Gesamt-Suite anpassen zu müssen. Insbesondere für Start-ups ist dieser Vorteil von hoher Bedeutung, da diese somit in der Lage sind, Unternehmenssoftware flexibel an wachsenden Geschäftsanforderungen anzupassen.

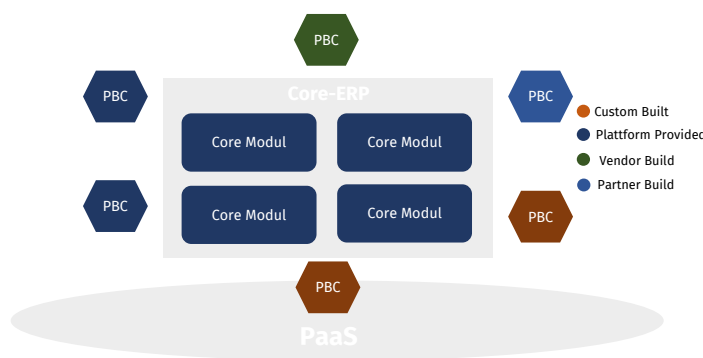


Abbildung 3: Bestandteile eines Composable-ERP-Systems. Eigene Darstellung.

Ein Composable-ERP-System verfügt über eine Kollektion von Kernkomponenten (s. Abb. 3)[23, S. 7]. Diese Kernkomponenten werden i.d.R. von einem einzigen ERP-Anbieter bereitgestellt und können deshalb ohne hohen Aufwand integriert

und aufeinander abgestimmt werden [22, S. 29 ff.]. Bei diesen Komponenten handelt es sich dabei um Funktionalitäten, welche das Hauptgeschäft eines Unternehmens unterstützen. Um den sich ändernden Geschäftsanforderungen gerecht zu werden, können diese Kernkomponenten durch zusätzliche PBCs erweitert werden [16, S. 58]. Dieses Konzept basiert auf dem Prinzip des *offenen Ökosystems*. Die externen in das Kern-ERP-System integrierbaren PBCs, umfassen dabei i.d.R. stark spezialisierte Funktionalitäten. So könnte sich ein E-Commerce-Unternehmen dazu entschließen eine Kern-Customer-Relationship-Management-Komponente (Kern-CRM-Komponenten) um eine KI-basierte Kundenanalysefunktion zu erweitern. Um die Integration dieser modularen Komponenten zu erleichtern, stellen ERP-Anbieter auf ihren Cloud-Plattformen einen zur Bündelung der PBCs verwendeten Marktplatz zur Verfügung. Die dabei angebotenen Komponenten können zusätzliche Funktionen des ERP-Kernanbieters oder externe Komponenten von Spezialherstellern darstellen. Auf diese Weise haben Mitarbeiter oder Teams die Möglichkeit, innerhalb ihres Unternehmens auf diesen Marktplatz zuzugreifen und Werkzeuge (PBCs), welche zur Unterstützung der operativen Tätigkeiten benötigt werden, ohne hohen Aufwand zu aktivieren. Das ERP-System kann somit auf die spezifischen Bedürfnisse und Anforderungen der Unternehmen zugeschnitten werden [22, S. 29 ff.]. IT-Systeme dienen dem Zweck der Unterstützung operativer Aufgaben. Um eine anwenderzentrierte Gestaltung der auf dem Marktplatz angebotenen Werkzeuge zu ermöglichen, sollte der Fokus dieser Tools auf den Bedürfnissen und Erwartungen der Anwender liegen (*Businesszentriertheit*). Deshalb ist essenziell, dass Mitarbeiter Systeme intuitiv nutzen und ggf. weiterentwickeln und anpassen können, ohne dabei von IT-Abteilungen abhängig zu sein [0]. Dabei soll die Verwendung von Low-Code/No-Code-Plattformen Abhilfe schaffen. Diese ermöglichen den Mitarbeiter eigene PBCs zu entwickeln ohne auf spezielle IT-Kenntnisse oder -Ressourcen angewiesen zu sein. Damit wird die Agilität und Flexibilität der CEs erhöht, während die Abhängigkeit von IT-Abteilungen reduziert wird.

2.1.2 Technologische Konzepte des Composable-Enterprises

Um die in Kapitel 2.1.1 aufgeführten betriebswirtschaftlichen Grundsätze in das Unternehmen zu integrieren, benötigt es verschiedener technologischer Konzepte. Zusammenfassen lassen sich diese mit dem Akronym *MACH*: *Microservices*, *APIs*, *Cloud-native*, *Headless*.

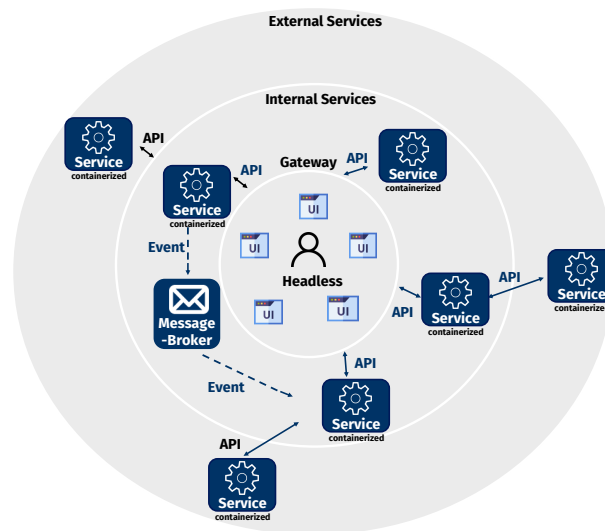


Abbildung 4: Technische Realisierung der Composable-Enterprise-Architektur. Eigene Darstellung.

Der zentrale Einstiegspunkt in eine CE-Anwendung ist das User-Interface. Insbesondere für Content-Management-Systeme wird für Frontend-Entwicklungen das *Headless-Konzept* verwendet (s. Abb. 4). Dieses beschreibt, dass zwischen Front- und Backend keine feste Kopplung besteht. Vielmehr werden auf dem Backend standardisierte Daten verwaltet, welche auf verschiedenen Frontends ausgegeben werden können [0]. Die Geschäftsfunktionen einer CEA werden in verschiedene PBCs gekapselt. In der Applikationslogik des Backends kann ein PBCs durch ein oder mehrere *Microservices* dargestellt werden. Ein Microservice ist eine eigenständige Einheit, welche für eine spezifische Funktion zuständig ist. Im Kontext eines E-Commerce-Unternehmens könnte ein PBC etwa eine Bestellverwaltung sein. Diese Bestellverwaltung kann dabei aus verschiedenen Microservices, wie einem Bestellannahme-, Auftragsabwicklungs- oder Rechnungstellungsdienst bestehen. Microservices spielen dabei hauptsächlich für die technische Realisierung eine Rolle. Folglich sind

diese für den Endanwender nicht erkennbar. Es ist möglich für jeden Service eine unterschiedliche Programmiersprache sowie Datenbank zu verwenden. So können Architektur und Technologien eines Dienstes unmittelbar an dessen betriebswirtschaftliche Anforderungen angepasst werden [16, S. 41]. Zur Kommunikation zwischen Services werden standardisierte Schnittstellen, sog. *Application-Programming-Interfaces (APIs)* verwendet. Mit APIs werden die von den Services bereitgestellten Funktionalitäten und Daten veröffentlicht [2, S. 15]. Diese Schnittstellen können dabei unmittelbar von dem Frontend oder anderen Microservices konsumiert werden. Ein weiteres in CEs verwendetes Kommunikationskonzept ist die *Event-driven Architecture (EDA)*. Die EDA ist ein Architekturkonzept, bei welchen Microservices asynchron über eine zentrale Vermittlungsinstanz, dem *Message Broker*, kommunizieren [3, S. 54]. Alle Komponenten der CEA werden auf einer Cloud-Plattform betrieben (*Cloud-native*). Cloud-Computing ist ein Dienstleistungsmodell, welches Nutzern ermöglicht Ressourcen, wie Speicher, Analyse-Tools oder Software über das Internet von einem Cloud-Anbieter zu beziehen [20, S. 5]. Dieses Computing-Modell ermöglicht IT-Services schnell und kosteneffizient an aktuelle Markterfordernisse anzupassen. Aufgrund der nutzungsabhängigen Bepreisung von Cloud-Plattformen können Dienste ohne hohen Investitionseinsatz auf- und abgebaut werden [20, S. 10]. Für das Cloud-Computing werden durch das National Institute of Standards and Technology (NIST) verschiedene Servicemodelle definiert. Neben *Software-as-a-Service (SaaS)* und *Infrastructure-as-a-Service (IaaS)*, bei welchem eine Anwendung bzw. eine gesamte Infrastruktur in der Cloud gemietet wird, gibt es ebenfalls das *Platform-as-a-Service (PaaS)* [20] [20, S. 9]. Bei diesem Computing-Modell wird eine Plattform bereitgestellt, auf welcher Kunden eigene Anwendungen entwickeln, testen und betreiben können. Ein auf dieser Service-Ebene von der SAP bereitgestelltes Produkt ist die SAP Business Technology Plattform (SAP BTP). Diese stellt eine Reihe von Diensten und Funktionen zur Verfügung, mit welchen Unternehmen SAP-ERP-Systeme anpassen, integrieren und erweitern können.

2.2 Integration und Bereitstellung einer Cloud-Anwendung

2.2.1 Agile und DevOps als moderne Softwareentwicklungskonzepte

Das Hauptaugenmerk eines CEs besteht darin eine möglichst modulare und flexible Systemarchitektur zu schaffen. Damit soll sichergestellt werden, dass IT-Leistungen in einem sich stetig ändernden Umfeld schnell und risikoarm bereitgestellt werden. Das traditionelle Wasserfallmodell, welches eine sequenzielle Abfolge der Projekt-elemente *Anforderung*, *Design*, *Implementierung*, *Test* und *Betrieb* vorgibt, besitzt dabei signifikante Limitationen. Die in dieser Methodik detailliert durchgeführte Vorabplanung, kann insbesondere in umfangreichen Langzeitvorhaben aufgrund unvorhersehbarer Externalitäten selten eingehalten werden [27, S. 5]. Dies resultiert nicht nur einem Anstieg der Kosten, sondern führt ebenfalls dazu, dass IT-Projekte länger als geplant ausfallen [26, S. 41]. Als Reaktion haben sich innerhalb der Projektmanagementlandschaft zunehmend **agile Vorgehensmodelle** etabliert. Im Gegensatz zum Wasserfallmodell, welches eine umfassende Vorabplanung vorsieht, wird das Vorhaben in einer agilen Entwicklung in viele zyklische Einheiten, sog. *Sprints*, segmentiert (s. Abb. 5) [10, S. 87]. Alle innerhalb des Projektumfangs zu entwickelnden Funktionalitäten werden dabei in einem zentralen Artefakt (*Product Backlog*) festgehalten und von dem Produktverantwortlichen (*Product Owner*) priorisiert.

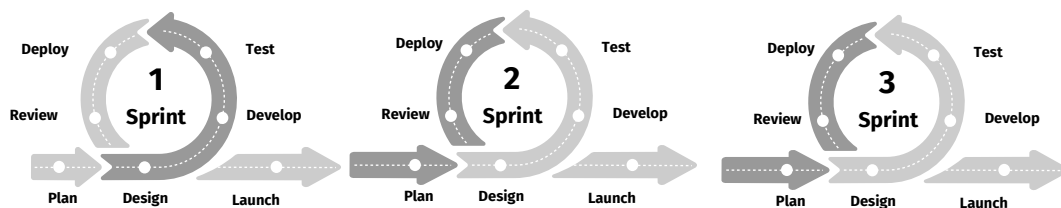


Abbildung 5: Exemplarische Abfolge eines agilen Entwicklungszykluses. In Anlehnung an K&C [0].

Sprints sind Durchläufe, welche i.d.R. einen Zeitraum von ein bis vier Wochen umfassen. Während dieses Abschnitts ist die Fertigstellung einer vor dem Sprint definierten Aufgabenkontingente (*Sprint Backlog*) vorgesehen. Nach Abschluss eines Sprints soll dabei ein potenziell an den Kunden auslieferbares Produkt zur Verfügung gestellt werden. Dies erlaubt eine schnelle Bereitstellung funktionsfähiger Software. Nach

Ablauf eines Sprints kann das an die Stakeholder ausgelieferte Artefakt als Feedback-Grundlage verwendet und im unmittelbaren Folge-Sprint eingearbeitet werden [0, S. 39]. Innerhalb der letzten Dekade haben sich diverse auf agilen Prinzipien basierenden Vorgehensmodelle, wie Scrum, Kanban oder Extreme Programming (XP) in der Softwareentwicklung etabliert. Obwohl einige dieser Methoden zur erfolgreichen Zusammenarbeit innerhalb der Entwicklungsteams beigetragen haben, bleibt das sog. *Problem der letzten Meile* bestehen [0]. Traditionell erfolgt eine funktionale Trennung der Entwickler- und IT-Betriebs-Teams. Das Problem der letzten Meile beschreibt dabei, dass aufgrund ausbleibender Kooperation dieser Teams der Programmcode nicht auf die Produktivumgebung abgestimmt ist. Erkenntnisse aus der Praxis zeigen, dass solche organisatorischen Silos häufig in einer schlechten Softwarequalität und somit in einem geminderten Ertragspotenzial bzw. in einer Erhöhung der Betriebskosten resultieren [11, S. 1]. So geht aus der von McKinsey veröffentlichten Studie *The Business Value of Design 2019* hervor, dass durchschnittlich 80 Prozent des Unternehmens-IT-Budgets zur Erhaltung des Status quo, also zum Betrieb bestehender Anwendungen verwendet wird. Stattdessen fordert das Beratungshaus eine Rationalisierung der Bereitstellung von Software, um finanzielle Mittel für wertschöpfende Investitionen zu maximieren [0]. Abhilfe schaffen kann das in der Literatur als **Development & Operations (DevOps)** bekannte Aufbrechen organisatorischer Silos zwischen Entwicklung und dem IT-Betrieb [11, S. 1]. Dabei stellt DevOps keine neue Erfindung dar. Stattdessen werden einzelne bereits bewährte Werkzeuge, Praktiken und Methoden, wie z.B. die agile Softwareentwicklung, zu einem umfassenden Rahmenwerk konsolidiert. DevOps zielt auf eine Optimierung des gesamten Applikationslebenszykluses, von Planung bis Bereitstellung der Software, ab. Während die agile Softwareentwicklung ausschließlich eine schnelle Lieferung von Software bezweckt, soll mit DevOps darüber hinaus sichergestellt, dass die Software effektiv betrieben wird. Aus diesem Grund gehören die Verwendung von Automatisierungstools, die Schaffung Feedbackschleifen zur Verbesserung der Softwarequalität sowie die Nutzung von Metriken zur Überwachung der Systemleistung zu den typischen Praktiken im Bereich DevOps. Mit diesen Methoden wird

eine enge Verzahnung von Entwicklung und Betrieb angestrebt, was dazu beiträgt, dass Teams Anwendungen zuverlässiger bereitstellen. (s. Anhang A.1).

2.2.2 Automatisierung der Integrations- und Bereitstellungsprozesse

Ein integraler Bestandteil des DevOps-Rahmenwerks ist *CI/CD*. CI/CD ist ein Verfahren, welches zur Verbesserung der Qualität bzw. zur Senkung der Entwicklungszeit von IT-Services beiträgt. Abhilfe schaffen soll dabei eine sog. CI/CD-Pipeline, welche alle Schritte von Code-Integration bis Bereitstellung der Software automatisiert. Hauptaugenmerk liegt dabei auf einer zuverlässigen und kontinuierlichen Bereitstellung von Software [28, S. 471].

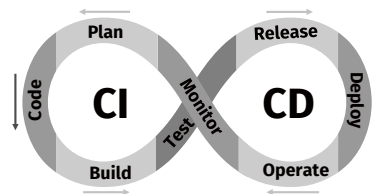


Abbildung 6: Aktivitäten im CI/CD-Prozess. In Anlehnung an Synopsys [0].

Alle in diesem Prozess anfallenden Aktivitäten werden dabei im CI/CD-Zyklus der Abb. 6 dargestellt. Der CI-Prozess (Continuous-Integration-Prozess) bezweckt, dass lokale Quellcodeänderungen in kurzen Intervallen und so schnell wie möglich in eine zentrale Codebasis geladen werden. Das frühzeitige Integrieren von Code soll dabei zu einer unmittelbaren und zuverlässigen Fehlererkennung innerhalb des Entwicklungsvorhabens beitragen [28, S. 471]. Der erste Schritt des CI-Prozesses umfasst die Planung zu entwickelnder Services (*Plan*: s. Abb. 6). Dabei soll festgestellt werden, welche Anforderungen eine Lösung besitzt bzw. welche Softwarearchitekturen sowie Sicherheitsmaßnahmen implementiert werden sollten. Um sicherzustellen, dass die in der Planung entworfene Anwendungsarchitektur auf das Design des Produktsystems abgestimmt ist, sollte zu jedem Zeitpunkt das Know-how der Betriebsteams einbezogen werden [11, S. 16]. Nach erfolgreichem Entwurf zu implementierender Anwendungsfeatures beginnt die Entwicklung der IT-Services (*Code*: s. Abb. 6). Arbeiten hierbei mehrere Entwickler parallel an demselben IT-Service, wird der

entsprechende Quellcode in Versionsverwaltungssysteme (*Repositories*) wie Github oder Bitbucket ausgelagert. Ein Repository stellt dabei einen zentralen Speicherort dar, welcher das Verfolgen sowie Überprüfen von Änderungen und ein paralleles bzw. konkurrierendes Arbeiten an einer gemeinsamen Codebasis ermöglicht [15, S. 31]. Der in dem Repository archivierte Hauptzweig (*Master-Branch*) enthält eine aktuelle und funktionsfähige Version des Codes. Dieser mit verschiedenen Validierungsprozessen überprüfte Code, stellt dabei die aktuelle in dem Produktionssystem laufende Anwendungsversion dar (s. Abb. 7). Im Sinne der agilen Entwicklung werden große Softwareanforderungen (*Epics*), in kleine Funktionalitäten (*User Stories*) segmentiert, welche in separate Feature-Branches des Repositories ausgelagert werden. Diese sind unabhängige Kopien des Hauptzweiges, in welcher ein Entwickler Änderungen vornehmen kann, ohne Konflikte in der gemeinsamen Codebasis zu verursachen. Nach Fertigstellung der Funktionalitäten sollte der um die Features erweiterte Quellcode so schnell wie möglich in den Hauptzweig integriert werden [15, S. 169].

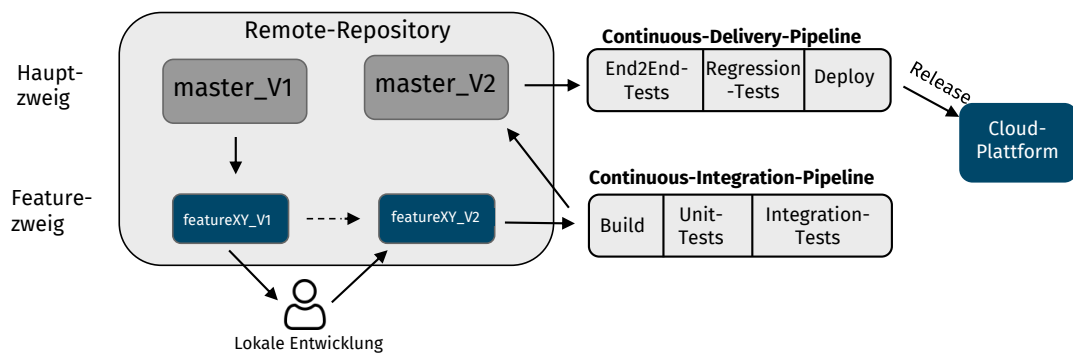


Abbildung 7: Integration der CI/CD-Pipeline mit dem Versionskontrollsystem. Eigene Darstellung.

Die Einbindung des Feature-Banches in den Hauptzweig resultiert i.d.R. in einem unmittelbaren und automatisierten Start des **CI/CD-Pipeline-Prozesses**. Bei der CI/CD-Pipeline handelt es sich dabei um ein vom Repository unabhängiges Bereitstellungsautomatisierungs-Tool, welche auf einer virtuellen Maschine oder in einer containerisierten Computing-Umgebung betrieben wird [14, Kap. 1.2]. Im ers-

ten Schritt des Pipeline-Prozesses wird die Applikationen zu einem ausführbaren Programm kompiliert (*Artefakt*) (*Build*: s. Abb. 6). Dafür können je nach Programmiersprache verschiedene Build-Tools, wie NPM für JavaScript oder Make für Multi-Target Application (MTA) verwendet werden [14, Kap. 7.1]. Nach Ablauf der Build-Workflows erfolgt eine automatische Abwicklung des Validierungsprozesses (*Smoke-Tests*).

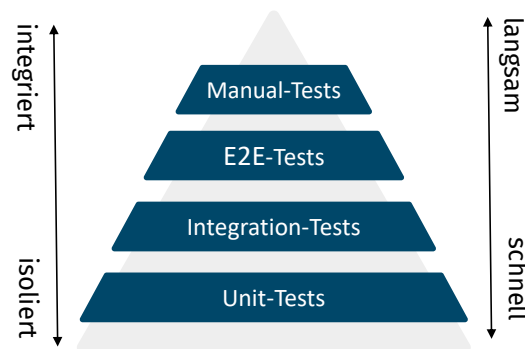


Abbildung 8: Hierarchische Darstellung von Softwaretests.
In Anlehnung an Paspelava [0].

Damit soll sichergestellt werden, dass zu jeder Zeit ein rudimentär getesteter Code bereitsteht und grundlegende Funktionalitäten sowie Schnittstellen erwartungsgemäß ausgeführt werden [11, S. 19]. Die in diesem Schritt abgewickelten Tests leiten sich dabei aus der *Definition of Done (DoD)* ab. Die DoD ist eine in der Planungsphase festgelegte Anforderungsspezifikation, deren Erfüllung als notwendige Voraussetzung für den Abschluss eines Features gilt. Somit sind Entwickler dazu angehalten, für jedes implementierte Feature einen der DoD entsprechenden Test zu entwerfen (*Test Driven Development*). Der in dem CI-Prozess bereitgestellte Code wird dabei überwiegend anhand schnell durchführbarer Tests überprüft. Der Zweck dieser zügigen Validierungen liegt dabei insbesondere darin, dass Entwickler zeitnahes Feedback auf die Erweiterungen erhalten. So können Fehler und Konflikte so schnell wie möglich entdeckt und behoben werden, was die Entwicklung bei einer reibungslosen Auslieferung der IT-Services unterstützt. Die in der CI-Pipeline abgewickelten Validierungen umfassen i.d.R. *Unit-* sowie *Integration-Tests* [14, Kap.

1.2]. Unit-Tests befinden sich dabei auf unterster Hierarchieebene der Test-Pyramide (s. Abb. 8). Somit besitzen diese eine kurze Ausführungsdauer, werden jedoch ausschließlich in einer isolierten Testumgebung abgewickelt. Mit Unit-Tests wird die funktionale Korrektheit kleinster Einheiten, wie z.B. Methoden einer Klasse, überprüft. Der Zweck der Unit-Tests besteht dabei in einer von externen Einflüssen und Daten unabhängigen Überprüfung der einzelnen Komponenten [12, Kap. 2]. Um bei der Bereitstellung neuer Funktionalitäten ebenfalls das Zusammenspiel verschiedener Komponenten zu überprüfen, werden *Integration-Tests* durchgeführt. Bei diesen Tests können Aspekte, wie der Austausch eines Nachrichtenmodells zweier Web-Services oder das Response-Objekt einer Datenbankabfrage untersucht werden [12, Kap. 2]. Im CI-Prozess werden i.d.R. auch einfache Code-Analysen durchgeführt. Diese sollen dem Entwickler eine schnelle Rückmeldung bezüglich Verletzung von Qualitätsstandards, potenziellen Schwachstellen sowie Leistungsproblemen liefern. Nachdem alle Tests erfolgreich absolviert wurden, kann sichergestellt werden, dass der neue Quellcode stabil, also funktionsfähig ist und keine Konflikte mit dem aktuellen Code des Hauptzweiges aufweist. Somit werden alle validierten Änderungen automatisch in dem Hauptzweig zusammengeführt. Mit diesem Prozessschritt beginnt der **Continuous-Delivery-Workflow (CD-Workflow)**. Während CI den Prozess der kontinuierlichen Integration des Quellcodes in das zentrale Repository verwaltet, steuert der CD-Workflow die Automatisierung der Anwendungsbereitstellung. Applikationen sollen somit ohne große Verzögerungen in die Produktivumgebung und somit zum Kunden ausgeliefert werden. Im Sinne des DevOps-Rahmenwerkes wird der CD-Prozess automatisch und unmittelbar nach Ablauf aller CI-Aktivitäten angestoßen. In der Praxis wird hierbei jedoch häufig ein manueller Schritt zwischengeschaltet [11, S. 20]. Damit soll sichergestellt werden, dass das Ausrollen der Anwendung erst nach Überprüfung und Genehmigung der Product Owner beginnt. Im ersten Schritt des CD-Prozesses wird das in die Produktivumgebung bereitzustellende Artefakt über die *Deployment-Pipeline* in eine *Staging-Area* geladen. Bei der Staging-Area handelt es sich dabei um ein System, welches zwischen Entwicklungs- und Produktivumgebung liegt. Die Staging-System-Konfigurationen werden dabei so ange-

legt, dass diese der Produktionsumgebung möglichst ähnlich sind [14, Kap. 1.3]. Neben Datenbanken werden hierbei ebenfalls Serverkonfigurationen, wie Firewall- oder Netzwerkeinstellungen von dem Produktivsystem übernommen. Somit soll sichergestellt werden, dass eine neue Anwendungsversion unter produktionsähnlichen Bedingungen getestet wird. Analog zum CI-Prozess werden innerhalb des CD-Workflows ebenfalls Unit- und Integration-Tests abgewickelt. Im Gegensatz zur CI-Pipeline werden dabei ebenfalls rechenintensive Tests automatisiert. Somit werden im CD-Prozess essenzielle, jedoch während des Entwicklungsworkflows zu aufwendige Validierungen durchgeführt [11, S. 20]. Darüber hinaus werden in der Staging Area ebenfalls in der Test-Pyramide (s. Abb. 8) höher positionierte, also rechenintensivere Tests ausgeführt [12, Kap. 2]. Dazu gehören *End-to-End-Tests (E2E-Tests)*. Mit diesen soll sichergestellt werden, dass die Anforderungen aller Stakeholder erfüllt werden. Hierbei wird ein vollständiges Anwenderszenario von Anfang bis Ende getestet. Dabei wird i.d.R. eine Benutzeroberfläche emuliert mit welcher etwa das Anmelden mit Benutzername, das Suchen eines Produktes und das Abschließen einer Bestellung validiert werden kann [0]. Für kritische Systeme werden während des Delivery-Prozesses ebenfalls *Regression-Tests* vorgenommen. Diese umfassen ein erneutes Testen bereits ausgelieferter Software-Komponenten. Regression-Tests können dabei in Form von Unit-, Integration- sowie Functional-Tests ausgeführt werden. Somit soll sichergestellt werden, dass sich Quellcodeänderungen nicht negativ auf die stabile Anwendungsversion auswirkt. Auf oberster Ebene der Test-Pyramide befinden sich die *Manual-Tests*. Dabei handelt es sich um von menschlichen Testern ausgeführte Validierungen, mit welchen Benutzerfreundlichkeit sowie Funktionalität anhand authentischer Anwenderszenarien gewährleistet werden soll. Da diese Tests nicht automatisiert durchgeführt werden können, muss der nächste Schritt der CD-Pipeline nach erfolgreicher Validierung manuell angestoßen werden. Im Anschluss werden i.d.R. verschiedene Codeanalysen abgewickelt. Hierbei werden Metriken, wie die prozentuale Testabdeckung oder Schwachstellen verwendeter Code-Patterns überprüft. Nach Durchführung der Codeanalysen wird das überprüfte Artefakt auf die Cloud-Plattform geladen (*Deploy*: s. Abb. 6). Je nach Bereitstellungsstrategie (s. 2.2.3),

wird die Anwendung dann unmittelbar oder erst nach weiteren Überprüfungen für den Kunden zugänglich gemacht. Der letzte Schritt des CD-Workflows umfasst die Laufzeitüberwachung der inbetriebgenommenen Anwendung (*Monitoring*: s. Abb. 6). Diese wird i.d.R. durch ein unabhängiges Überwachungssystem und nicht von dem Pipeline-Tool selbst abgewickelt. Dabei können z.B. Dashboards zur Analyse der Build-, Test- und Deployment-Prozesse visualisiert werden. Darüber hinaus umfasst dieses Tool essenzielle Überwachungselemente zum *Infrastruktur-, Plattform- sowie Anwendungs-Monitoring*. Beim Infrastruktur-Monitoring werden Metriken wie CPU-, Speicher- und Netzwerklast der Server bzw. Datenbanken untersucht. Das Plattform-Monitoring setzt dabei eine Ebene höher an und validiert, dass Komponenten wie Datenbanken, virtuelle Netze bzw. Middlewares ordnungsgemäß durchgeführt werden. Das Anwendungs-Monitoring umfasst die Überwachung der Funktionalitäten und der Applikation selbst [11, S. 21].

Zur Automatisierung der CI/CD-Prozesse werden von der SAP i.d.R. drei verschiedene Tools vorgeschlagen. Eine unmittelbare von der SAP bereitgestellte Lösung ist das *SAP Continuous Integration and Delivery (SAP CI/CD)*. Das SAP CI/CD ist eine auf der SAP BTP betriebene SaaS-Lösung, mit welcher vordefinierte Pipeline-Templates konfiguriert und ausgeführt werden können. Dieses Tool ist insbesondere mit SAP-Standardtechnologien, wie den Programmierframeworks SAP UI5 und SAP CAP Node sowie der Laufzeitumgebung Cloud-Foundry kompatibel [0]. Eine weitere bei der SAP empfohlene CI/CD-Alternative ist das Open-Source-Tool *Jenkins*. Im Gegensatz zum templatebasierten SAP CI/CD, muss der Bereitstellungsworkflow bei Jenkins mit der Programmiersprache Groovy implementiert werden. Weiterhin wird Jenkins nicht unmittelbar auf der SAP BTP betrieben, sondern muss auf einem eigenen Server (On-Premise) verwaltet werden [14, Kap. 2]. Um die Bereitstellung von SAP-spezifischen Technologien zu optimieren, wurde die Programmbibliothek *Project Piper* veröffentlicht. In Project Piper werden vorimplementierte Pipeline-Schritte gebündelt. Im Gegensatz zum SAP CI/CD sind diese jedoch hoch konfigurierbaren. Ein externes ebenfalls von der SAP vorgeschlagenes CI/CD-Werkzeug ist *Azure Pipelines*. Azure Pipelines ist ein von Microsoft entwickeltes Tool, welches

umfassende Integrationsmöglichkeiten zu anderen Microsoft-Diensten wie die Azure-Cloud-Plattform oder Microsoft Visual Studio Code bietet. Zur Implementierung des CI/CD-Workflows SAP-spezifischer Technologien wird für Azure Pipelines ebenfalls die Programmbibliothek Project-Piper verwendet.

2.2.3 Strategien zur Bereitstellung von Neuentwicklungen

Nachdem das Artefakt auf einer virtuellen Maschine bzw. containerisierten Cloud-Instanz installiert und gestartet wurde, erfolgt die Inbetriebnahme der neuen Anwendungsversion je nach Bereitstellungsstrategie unmittelbar oder erst nach weiteren Validierung. Anhand der Bereitstellungsstrategie wird festgelegt, mit welcher Methode und zu welchem Zeitpunkt Nutzeranfragen von der aktuellen auf die neue Anwendungsinstanz umgeleitet werden.

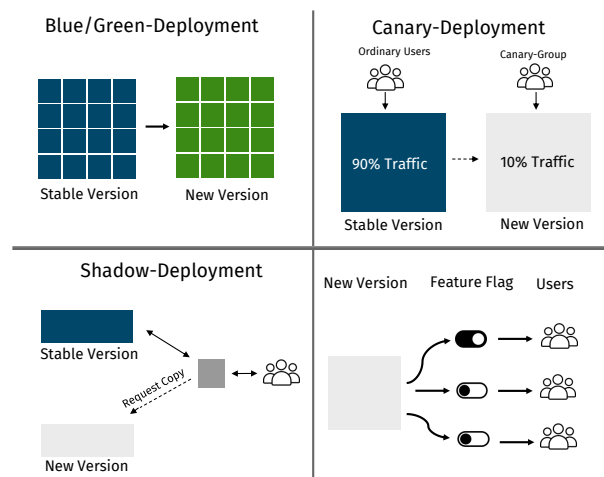


Abbildung 9: Strategien zur Bereitstellung von Software.
In Anlehnung an Ugochi [0].

Eine häufig verwendete Deployment-Strategie ist dabei das *Blue/Green-Deployment*. Hierbei wird neben der stabilen aktuellen Anwendung (*Blaue Version*) ebenfalls eine Instanz der neuen Anwendung (*Grüne Version*) betrieben. Nutzeranfragen werden dabei von dem Lastenverteilungsservice (*Load-Balancer*) erst nach Validierung aller Tests umgeschaltet. Dazu gehören neben den in der CI/CD-Pipeline definierten Tests ebenfalls Überprüfungen der Qualitätssicherung. Diese umfassen manu-

elle Tests, in welchen Funktionen, Benutzeroberfläche sowie die Anwenderfreundlichkeit überprüft werden [0]. Im Gegensatz zum Blue/Green-Deployment, bei welchem eine neue Version simultan für die gesamte Nutzerbasis zur Verfügung gestellt wird, gewährleistet das *Canary-Deployment* eine restriktivere Nutzlastumleitung. Hierfür wird die neue Anwendungsversion vorerst einer überschaubaren Nutzeranzahl (*Canary-Gruppe*) bereitgestellt. Dabei sollte die zusammengestellte Canary-Gruppe die Gesamtnutzerbasis möglichst gut repräsentieren. Anhand des Canary-Traffics soll der fehlerfreie Betrieb neuer Anwendungen überprüft und ggf. Anpassungen vorgenommen werden. Bevor die neue Anwendungsversion der gesamten Nutzerbasis zur Verfügung gestellt, kann diese sukzessive und schrittweise ausgerollt werden [0]. Eine aufwendigere, jedoch risikoärmere Bereitstellungsstrategie stellt das *Shadow-Deployment* dar. Dabei wird neben der Instanz der aktuellen Version ebenfalls ein sog. *Shadow-Model* auf der Infrastruktur betrieben. Das Shadow-Model verwaltet die neue Version der Anwendung, kann jedoch nicht unmittelbar von den Nutzern aufgerufen werden. Diese Instanz stellt ein hinter der stabilen Version gelagertes Schattenmodell dar. Benutzeranfragen werden von dem Load-Balancer stets auf die aktuelle Version der Instanz weitergeleitet, verarbeitet und beantwortet. Gleichzeitig wird eine Kopie dieser Anfrage an das Shadow-Model weitergeleitet und von diesem prozessiert. Die Shadow-Modell-Verarbeitung des in der Produktionsumgebung abgewickelten Netzwerkverkehrs ermöglicht den Entwicklern somit eine anwendungsbezogene Überprüfung entwickelter Features [0]. Ein weiteres Bereitstellungskonzept sind *Feature-Flags*. Bei dieser Methode wird die Sichtbarkeit neuer Funktionalitäten an eine *Flag* gekoppelt. Diese Flags repräsentieren globale Variablen, welche von dem Systemadministrator oder den Anwendern gesetzt werden können. Abhängig von dem Zustand der Flag kann gesteuert werden, ob die im CI/CD-Prozess bereitgestellten Features für Anwender sichtbar sind [0].

3 Methodische Vorgehensweise

Der Forschungsbereich dieser wissenschaftlichen Abhandlung umfasst die Themengebiete CI/CD sowie CEA. Da in Kombination dieser beiden Forschungsbereiche sowie in der praktischen Umsetzung dieser Konzepte mit SAP-spezifischen Technologien in der Literatur kein Datenmaterial vorhanden ist, werden im Rahmen dieser Arbeit Experteninterviews durchgeführt. Zur abschließenden Analyse und Bewertung der CI/CD-Tools wird das AHP-Verfahren angewandt. Dabei sollen die in den Gesprächen erhobenen Daten als Entscheidungsgrundlage zur Durchführung des Analyseverfahrens verwendet werden.

3.1 Semistrukturierte Leitfadeninterviews zur Erhebung qualitativer Daten

Das Experteninterview stellt eine häufig angewandte Analysemethode dar, welche vorrangig bei qualitativen Untersuchungen verwendet wird. Die Meinungen, Erfahrungen und Perspektiven der Experten werden dazu verwendet, relevante Aspekte zu einem Thema zu identifizieren oder eine Forschungshypothese zu formulieren. Diese wissenschaftliche Methode wird dabei insbesondere für aktuelle, stets unerforschte Themen sowie Fragestellungen mit geringem Literaturaufkommen verwendet [9, S. 363 ff.]. Als Experten werden in diesem Zusammenhang Interviewpartner bezeichnet, welche aufgrund ihres im Rahmen beruflicher Tätigkeiten erworbenen Wissens umfassende Kenntnisse in einem spezifischen Fachgebiet besitzen. In der Literatur werden verschiedene Arten von Experteninterviews definiert. Dazu gehören strukturierte, semistrukturierte sowie unstrukturierte Interviews [9, S. 363 ff.]. Strukturierte Expertengespräche zeichnen sich dabei insbesondere durch die Vorabfestlegung der im Interview gestellten Fragen aus. Hierbei wird bezweckt, dass allen Teilnehmenden dieselben Fragen in standardisierter Reihenfolge vorgelegt werden. Im anderen Extrem der unstrukturierten Interviews erfolgt lediglich eine Bestimmung des Gesprächsthemas, jedoch werden vorab keine expliziten Fragen festgelegt. Den konkreten Verlauf des Gespräches bestimmt dabei die dynamische Entwicklung des Antwort-

Nachfrage-Verhaltens der Interviewteilnehmer. Aufgrund des eng abgegrenzten Forschungsbereichs, besteht bei der Durchführung von unstrukturierten Interviews in dieser Arbeit das Risiko, dass die Gesprächsinhalte vom eigentlichen Untersuchungsgegenstand abweichen. Auch die Abwicklung von strukturierten Interviews ist für diese Arbeit nicht geeignet [13, S. 244 ff.]. Das liegt insbesondere an dem starren Erhebungsdesign der strukturierten Interviews. Angesichts der in dieser Arbeit angestrebten Erschließung unerforschter Themengebiete bietet eine Vorabfestlegung der Fragen nicht genügend Flexibilität, um auf neu auftretende Aspekte innerhalb des Gesprächs angemessen zu reagieren. Deshalb werden im Rahmen dieser Arbeit semistrukturierte Interviews durchgeführt. Diese Interviewform realisiert eine Leitfadenstruktur, welche eine vordefinierte Fragegestaltung vorgibt, jedoch im Verlauf des Gesprächs gleichzeitig ein hohes Maß an Flexibilität bietet. Ergebnisse während eines Expertengesprächs neue Aspekte, können diese mit unmittelbaren Ad-hoc-Fragen aufgegriffen werden. Um die Interviews auszuwerten, muss eine Transkription der Expertengespräche erfolgen [13, S. 244 ff.]. Da zur Beantwortung der Forschungsfrage dieser Arbeit nicht der exakte Wortlaut, sondern vielmehr die inhaltliche Ausgestaltung der Expertengespräche von Bedeutung ist, wird eine zusammenfassende Transkription durchgeführt. Zur Auswertung der Transkription wird i.d.R. eine deduktive bzw. induktive Methode verwendet. Bei einer deduktiven Auswertung werden Aussagen der Experten vordefinierten Kategorien zugeordnet [13, S. 244 ff.]. Dieses Evaluationsverfahren bietet insbesondere zur Validierung einer vorabdefinierten Forschungshypothese einen erheblichen Mehrwert. Da im Rahmen dieser Arbeit stattdessen die Beantwortung einer offenen Fragestellung vorgesehen ist, wird eine induktive Kodierung der Interviews vorgenommen. Statt Kategorien im Voraus festzulegen, werden diese bei der induktiven Kodierung dynamisch aus dem Interviewmaterial abgeleitet. Eine implizite Auswertung der induktiven Kodierung wird im AHP-Verfahren (s. Kap. 4) angewendet. So werden die während der Evaluation getroffenen Entscheidungen anhand der Expertenaussagen referenziert.

3.2 Evaluation von Integrations- und Bereitstellungs-Tools unter Anwendung des Analytischen Hierarchieprozesses

AHP ist ein von dem Mathematiker Thomas Saaty konzipiertes Entscheidungs-Framework. Dieses Rahmenwerk eignet sich insbesondere für komplexe betriebswirtschaftliche und technische Entscheidungsprobleme. Bei AHP wird eine Bewertung der Entscheidungsalternativen anhand verschiedener Kriterien vorgenommen. Da das zugrundeliegende Rahmenwerk auf der Prämisse einer divergierenden Wichtigkeit der unterschiedlichen Entscheidungskriterien basiert, muss für die festgelegten Kriterien eine Gewichtung vorgenommen werden. [21, S. 86]. Das AHP-Verfahren besteht dabei aus mehreren Schritten. Zu Beginn des AHP-Verfahrens ist eine exakte Definition der zu lösenden Problemstellung notwendig. Im nächsten Schritt werden verschiedene Entscheidungsalternativen sowie Bewertungskriterien festgelegt. Die Entscheidungsstruktur kann dabei beliebig durch einen hierarchischen Aufbau gestaltet werden (s. Abb. 11).

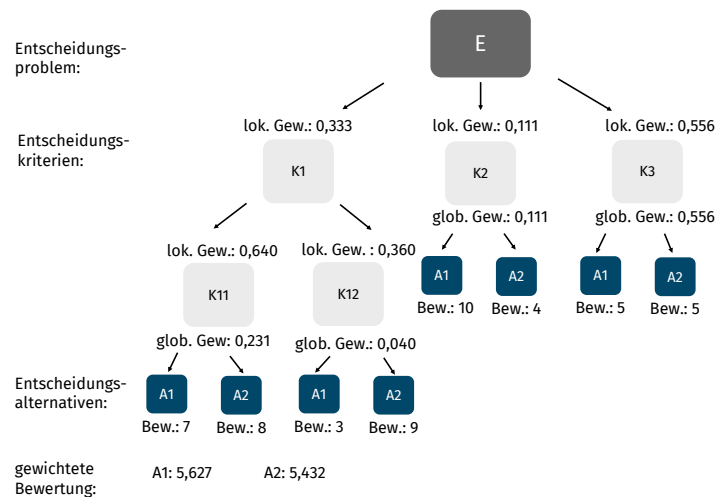


Abbildung 10: Exemplarische Darstellung der hierarchischen Entscheidungsstruktur im AHP. Eigene Darstellung.

Auf oberster Ebene des AHP-Baums befindet sich das Entscheidungsproblem. Es besteht die Möglichkeit, ein Kriterium in mehrere Subkriterien zu unterteilen. Auf unterster Ebene befinden sich schließlich die im Hinblick auf die festgelegten Evaluationskriterien zu bewertenden Entscheidungsalternativen. Um eine auf die Präferenzen

der Stakeholder abgestimmte Bewertung zu ermöglichen, müssen die zuvor festgelegten Entscheidungskriterien gewichtet werden. Besteht der AHP-Baum aus mehreren Stufen, erfolgt für jede Ebene zunächst eine isolierte Gewichtung. Zur Bestimmung der relativen Wichtigkeit ist für das AHP-Verfahren nach Saaty ein paarweiser Vergleich vorhergesehen. Hierbei wird die Wichtigkeit eines Kriteriums gegenüber eines Anderen ermittelt. Die in der Literatur aufgeführte Gegenüberstellung erfolgt dabei auf einer Skala von eins bis neun [21, S. 86]. Um den Prozess der Entscheidungsfindung für die Probanden intuitiver zu gestalten, wird von der in der Literatur etablierten Vorgehensweise abgewichen und eine eigene Methode konzipiert. So ist im Rahmen dieser Arbeit eine Gewichtung von null bis zwei vorhergesehen. Während der Wert zwei impliziert, dass ein Entscheidungskriterium wesentlich wichtiger ist, wird mit dem Index eins eine gleiche Gewichtung ausgedrückt. Eine relative Wichtigkeit mit dem Wert null bedeutet in diesem Zusammenhang, dass ein Kriterium weniger wichtig als die Vergleichskategorie ist.

	Kriterium K1	Kriterium K2	Kriterium K3	V_{ik}	lok. Gew. (W_{ik})
Kriterium K1	1	2	0	3	0,333
Kriterium K2	0	1	0	1	0,111
Kriterium K3	2	2	1	1	0,556

Abbildung 11: Exemplarische Darstellung der Paarvergleichsmatrix im AHP.
Eigene Darstellung.

So wird dem in Abb. 11 exemplarisch dargestellten Kriterium K1 eine wesentlichere Bedeutung als K2 zugeschrieben. Das korrespondierende Äquivalens auf der rechten Matrixhälfte besitzt entsprechend eine Gewichtung von null (*weniger wichtig*). Um die Zuverlässigkeit der Paarvergleichsgewichtungen zu gewährleisten ist für das AHP in der Literatur die Berechnung eines Konsistenzindex vorgesehen. Angesichts des hohen Zeitaufwands, welchen die Überprüfung transitiver Beziehungen bei der Durchführung der Gewichtung durch die Experten erfordern würde, wurde auf eine derartige Analyse im Rahmen dieser Arbeit verzichtet. Um der in der Paarvergleichsmatrix getroffenen Gegenüberstellung eine höhere Aussagekraft zu verleihen,

müssen die relativen Gewichtungen in prozentuale Werte transformiert werden. Im ersten Schritt werden dabei alle Paarvergleichswerte eines Entscheidungskriteriums aufsummiert:

$$V_{ki} = X_{KiKj} + X_{KiKj} + X_{KiKj}$$

Anschließend muss diese Summe standardisiert werden. Hierfür wird die Gewichtungssumme (V_{ki}) eines Kriteriums durch die Gesamtanzahl der in einer Paarvergleichsmatrix (N) vergebenen Punkte dividiert. (N) ergibt sich durch die Anzahl der Kriterien (n) einer Paarvergleichsmatrix:

$$N = n \cdot n$$

Anschließend kann die prozentuale Gewichtung eines Entscheidungskriteriums wie folgt berechnet werden:

$$W_{ki} = \frac{V_{ki}}{N}$$

Um die Zuverl Diese lokale Gewichtung wird für jede Hierarchieebene durchgeführt. Um die globale Gewichtung (P_{ki}) zu ermitteln, wird die Gewichtung jedes Subkriteriums mit den Gewichtungen der übergeordneten Kriterien multipliziert:

$$P_{ki} = p_1 \cdot p_2 \cdot p_3 \cdot p_n$$

In der Literatur wird vorgesehen, dass auf unterster Hierarchieebene ebenfalls eine Gewichtung der Entscheidungsalternativen vorgenommen wird, um eine Bewertung durchzuführen [21, S. 86]. Da dies insbesondere bei auf subjektiven Präferenzen basierenden Problemstellungen eine wichtige Rolle spielt, wird hierbei von dem Leitfaden nach Saaty abgewichen. Stattdessen wird für jedes Kriterium auf unterster Ebene eine feste Metrik definiert, anhand welcher die Alternativen bewertet werden. Letztlich werden die für jedes Kriterium erzielten Punkte mit den globalen Gewichtungsfaktoren multipliziert und alle Teilbewertungen zu einer gewichteten Gesamtbenotung zusammengezogen. Die Entscheidungsalternative mit der höchsten Gesamtbewertung gilt dabei als optimales Modell. Das Rahmenwerk eignet sich aufgrund des multidimensionalen Entscheidungsmodells besonders für die in der

Arbeit zu untersuchende Fragestellung. Die bei der Wahl einer CI/CD-Pipeline zu berücksichtigenden Aspekte können somit als Bewertungskriterien in den AHP-Baum aufgenommen werden. Des Weiteren ermöglicht die im AHP-Verfahren abgewickelte Gewichtung, dass die Kriterien in unterschiedlichem Maße Einfluss auf die Bewertung einer Pipeline nehmen. Im Rahmen dieser Arbeit kann somit die Wichtigkeit der an die Entscheidung geknüpften Aspekte durch verschiedene an der Bereitstellung von Software beteiligten Stakeholder (Entwickler, DevOps-Spezialisten etc.) festgelegt werden (s. Kap. 4.3). Dies ermöglicht eine auf die Präferenzen der Entscheidungsträger abgestimmte Bewertung der zu untersuchenden Pipelines. Das AHP-Verfahren besitzt darüber hinaus ebenfalls einen Vorteil bezüglich des Bewertungsdesigns. Während bei Methoden wie der SWOT-Analyse ausschließlich qualitative Aspekte berücksichtigt werden, ermöglicht das AHP-Model ebenfalls eine Evaluation quantitativer Bewertungsmetriken. Infolgedessen kann die Definition der Bewertungsmetrik flexibel und in Abhängigkeit des Entscheidungskriteriums getroffen werden. Im Rahmen dieser Arbeit wird anhand des AHP-Verfahrens ein für CEA optimales CI/CD-Tool bestimmt. Dabei ist jedoch zu beachten, dass bei der Wahl einer CI/CD-Pipeline ebenfalls K.O.-Kriterien existieren können, welche dazu führen, dass das Rahmenwerk für die Entscheidung keine relevante Aussagekraft mehr besitzt. Deshalb werden bei der Entwicklung der Handlungsempfehlung (s. Kap. 5) Szenarien definiert, bei welchen die Wahl der CI/CD-Pipeline vom AHP-Ergebnis abweicht. Weitere Erläuterungen zu den im AHP-Verfahren getroffenen Entscheidungen werden zur besseren Verständlichkeit in Kapitel 4 ausgeführt.

4 Evaluation der Integrations- und Bereitstellungs-Pipelines unter Anwendung des Analytischen Hierarchieprozesses

Als Entscheidungsalternativen werden CI/CD-Pipelines für die Bereitstellung von Cloud-Software gegenübergestellt. Konkret handelt es sich dabei, um die Tools *SAP CI/CD*, *Jenkins*, sowie *Azure Pipelines*. Bei der Untersuchung der Pipelines sind dabei folgende Rahmenbedingungen zu beachten: Im Rahmen der Arbeit soll evaluiert werden, ob sich die verschiedenen Pipelines zur Bereitstellung von Software für CEs eignen. Weiterhin muss festgestellt werden, inwiefern die Tools für die Technologien SAP CAP bzw. SAP UI5 kompatibel sind und ob eine Bereitstellung in der Laufzeitumgebung Cloud Foundry der SAP BTP möglich ist.

4.1 Definition der Entscheidungskriterien

Die zur Durchführung des AHP-Verfahrens benötigten Daten werden neben einer Literaturrecherche ebenfalls mittels Experteninterviews erhoben. Für die Experteninterviews wird ein Gremium aus acht Mitarbeitenden der SAP zusammengestellt (s. Anhang C). Diese sind jeweils in verschiedenen Bereichen der Cloud-Fullstack-Entwicklung, Test-Management, Product Management sowie in der SoftwareArchitektur spezialisiert. Somit kann Expertise über verschiedene Fachbereiche hinweg aufgebaut und Anforderungen aller an der Entwicklung, Bereitstellung sowie an dem Betrieb von Software beteiligten Stakeholdern erfasst werden. Zur Festlegung der Entscheidungskriterien wird eine induktive Kodierung der Expertengespräche durchgeführt (s. Anhang C.5). Dabei werden aus besonders häufig von Experten genannten Aspekten systematisch Kategorien abgeleitet. Diese umfassen insbesondere Aspekte, welche die in vergangenen Kundenprojekten hervorgegangenen Anforderungen an eine CI/CD-Pipeline widerspiegeln. Da innerhalb dieser Kundenprojekte oft weniger strikte Qualitätsanforderungen an den CI/CD-Prozess gestellt werden, wird darüber hinaus erarbeitet, welche internen Bestimmungen von der SAP zur Bereit-

stellung von Standardsoftware definiert werden. Die bei der induktiven Kodierung erhobenen Kategorien werden anschließend ebenfalls als Entscheidungskriterien im AHP-Verfahren wiederverwendet. Die mit dieser Vorgehensweise erhobenen Entscheidungsalternativen sind folgender Abbildung zu entnehmen:


	K1 Funktionalität
	K1.1 Tests K1.2 Code-Analysen K1.3 Build K1.4 Deploy K1.5 Monitoring
	K2 Integrationsmöglichkeiten
	K2.1 Integration in Repository K2.2 Integration in Entwicklungsumgebung K2.3 Integration in Planungs-Software
	K3 Kosten
	K4 Skalierbarkeit
	K5 Performance
	K5.1 Integration-Zeit K5.2 Delivery-Zeit
	K6 Flexibilität
	K7 Support
	K7.1 Administrativer Support K7.2 Community Support
	K8 Sicherheit
	K8.1 Authentifizierung und Autorisierung K8.2 Sicherheitsarchitektur
	K9 Benutzerfreundlichkeit
	K9.1 Installation und Wartung K9.2 Intuitive Bedienbarkeit

Abbildung 12: AHP-Entscheidungsstruktur zur Bewertung von CI/CD-Pipelines. Eigene Darstellung.

Auf der obersten Ebene des AHP-Entscheidungsbaums werden neun Kategorien definiert. Das erste Kriterium ist **Funktionalität** (K1). Diese Kategorie umfasst verschiedene innerhalb des CI/CD-Workflows benötigte funktionale Spezifikationen. So sollte eine Pipeline laut Experte 1 etwa dazu in der Lage sein, Anwendungen zu testen, Code-Analysen durchzuführen und eine Software auf der Cloud-Plattform bereitzustellen [29, Z. 52]. Angesichts der Vielfältigkeit des Entscheidungskriteriums Funktionalität wird eine Untergliederung in verschiedene Subkriterien vorgenommen. In Kriterium K1.1 wird dabei die Unterstützung automatisierter

Unit-Tests evaluiert [24, Z. 16]. Von der SAP werden diesbezüglich Produktstandards vorgegeben. Diese stützen sich auf *ISO 9001*, eine internationale Norm für Qualitätsstandards. Im Kontext der Softwareentwicklung verlangt diese, eine für neue Funktionalitäten kontinuierliche und automatisierte durchgeführte Prüfung [6, Z. 61]. Dies umfasst eine Abwicklung von Unit-, Integration-, E2E-Test sowie Regression-Tests für Backend sowie Frontend. Hinsichtlich der Entwicklung von CAP-Node-Anwendungen wird in der SAP die Durchführung von Unit-Tests mittels Jest bzw. Mocha und Integration-Tests mittels Newman vorgeschlagen. Für die Programmierung mit SAP UI5 sind hingegen Unit-Tests mittels Q-Unit, Integration-Tests mittels OPA5 und E2E-Tests mittels WDI5 vorgesehen [6, Z. 62 ff.]. Für Regression-Tests wird hingegen kein spezifisches Test-Framework verwendet. In diesem Zusammenhang wird lediglich überprüft, ob eine erneute Validierung bereits bestehender Software-Komponenten durchführbar ist. In Kriterium K1.2 wird die Kompatibilität verschiedener *Code-Analyse-Tools* untersucht [29, Z. 67 ff.]. Es wurde gezielt eine Trennung dieser Kategorie mit dem Kriterium K1.2 (Tests) vorgenommen. Während Tests eine funktionale Erfüllung der Anforderung evaluieren, werden mit den Analysen Code-Qualitätsstandards der entwickelten Funktionalitäten untersucht. Dabei wird evaluiert, ob statische Codeanalysen, Security- sowie Performance-Überprüfungen von den CI/CD-Pipelines unterstützt werden. Laut Experte 3 werden für statische Code-Analysen gemäß Produktstandards der SAP Lint sowie SonarQube verwendet [19, Z. 41]. Zur Durchführung von Sicherheitsüberprüfungen wird für SAP-CAP-Node-Anwendungen Checkmarx verwendet, während für SAP UI5 DASTER zum Einsatz kommt. Für Performance-Tests wird das Tool JMeter verwendet. In der Kategorie K1.3 werden die *Build-Funktionalitäten* der CI/CD-Pipelines evaluiert [29, Z. 67 ff.]. Dabei wird untersucht, ob die Entscheidungsalternativen das Build-Tool Make zum Kompillieren von Multi-Target-Applications (MTAs) unterstützen. Die MTA ist eine Applikation, z.B. ein Microservice eines CEs, welche aus verschiedenen Modulen besteht. Diese Module umfassen typischerweise die durch einen Microservice bereitgestellte API oder eine von der Applikation verwendete Datenbank. MTAs werden dabei i.d.R. für die Bereitstellung von

SAP-CAP-Node- sowie SAP-UI5-Anwendungen verwendetet [0]. Laut Experte 1 ist weiterhin essenziell, dass Pipelines Artefakt-Repositories unterstützen [29, Z. 37 ff.]. Eine MTA kann von verschiedenen externen Ressourcen (z.B. Bibliotheken) abhängig sein. Diese sind in einem Artefakt-Repository verwaltete externe Komponenten, welche bei der Entwicklung neuer Microservices wiederverwendet werden können [29, Z. 40]. Diese Komponenten werden während des Build-Prozesses von der CI/CD-Pipeline aus den Artefakt-Repositories geladen und kompiliert. Ein weiterer für CEs essenzieller Build-Aspekt ist die Unterstützung von Docker-Workflows. Durch den Einsatz von Docker-Containern können Entwickler schnell virtualisierte Umgebungen mit benötigten Frameworks und Tools bereitstellen, ohne dabei eine gesamte Infrastruktur manuell konfigurieren zu müssen [0]. In Kriterium K1.4 werden die *Deploy- und Release-Funktionalitäten* der CI/CD-Tools untersucht [29, Z. 68 ff.]. Dabei wird evaluiert, ob die Pipelines neben dem Ausrollen von Software auf die Cloud-Foundry-Laufzeitumgebung ebenfalls verschiedene Bereitstellungsstrategien unterstützen (z.B. Blue/Green-Deployment s. Kap. 2.2.3). Des Weiteren wird erörtert, ob mit den CI/CD-Pipelines eine Bereitstellung in das SAP Cloud Transport Management (SAP CTM) möglich ist. Das SAP CTM kann als zusätzliche Schicht im CI/CD-Prozess verbaut werden (s. Abb. 16). Experte 2 merkt an, dass dieses System dabei insbesondere innerhalb komplexer ERP-Landschaften zur Optimierung der Bereitstellungsprozesse führen kann (weitere Details in Kap. 4.4) [18, Z. 59]. In Kategorie K1.5 wird die *Monitoring-Funktionalität* der verschiedenen CI/CD-Pipelines untersucht [18, Z. 37 ff.]. In diesem Zusammenhang erfolgt eine Bewertung der Überwachbarkeit der CI/CD-Tools. Für interne Projekte wird dabei i.d.R. das SAP-Partner-Tool Splunk verwendet [18, Z. 70]. Damit lassen sich verschiedene Metriken, wie Build-Zeiten, Performance-Checks oder Fehlerquoten verschiedener Pipelines in einem zentralisierten Dashboard visualisieren. Da CI/CD-Pipelines im Rahmen dieser Arbeit ebenfalls für externe Kundenprojekte evaluiert werden, ist innerhalb dieses Kriteriums ebenfalls eine Betrachtung von externen Open-Source-Tools. Laut Experte 2 wird dabei häufig das Kibana-Dashboard verwendet [18, Z. 70]. In Kategorie K2 werden die **Integrationsmöglichkeiten** der Pipelines untersucht. In

dem Subkriterium *Integrationsmöglichkeiten von Repositories (Kategorie K2.1)* wird evaluiert, ob sich das Repository in die Pipeline integrieren lässt [29, Z. 79]. Damit können bestimmte Ereignisse, wie Push-Mitteilungen bei Code-Änderungen automatisiert an die CI/CD-Pipeline übermittelt werden. Somit kann ein unmittelbarer Integrations- bzw. Bereitstellungs-Workflow ausgelöst werden. Bei der Bewertung wird ein besonderes Augenmerk darauf gelegt, dass häufig verwendete Repositories problemlos in die Pipeline integrierbar sind. Da in der Literatur diesbezüglich keine öffentlichen Statistiken zugänglich sind, wird auf empirische Einschätzungen der Experten zurückgegriffen. So wurden nach Beurteilung des Experten 3 innerhalb interner und externer Projekte am häufigsten GitHub, GitLab und BitBucket verwendet [7, Z. 85]. In Kriterium K2.2 werden die *Integrationsmöglichkeiten von Entwicklungsumgebung* untersucht [29, Z. 81 ff.]. Die Integration-Pipeline kann unmittelbar während des Entwicklungsprozesses aus der Entwicklungsumgebung gestartet werden. Auf diese Weise wird sichergestellt, dass Entwickler Feedback in noch kürzeren Zeitabständen erhalten als bei einer ausschließlichen Integration der Pipeline in das Repository. Die Bewertung bezieht sich dabei ausschließlich auf SAP UI5 sowie SAP CAP Node Entwicklungsumgebungen. Dazu gehören Microsoft Visual Studio Code, SAP Business Application Studio (SAP BAS) sowie Eclipse. In Kriterium K2.3 wird die *Integrationsmöglichkeit von Planungs-Software* untersucht [7, Z. 89 ff.]. Dazu gehören Projektmanagement-Tools wie Jira. Laut Experte 4 ermöglicht eine Integration solcher Planungssoftware Projektmanager eine erhöhte Transparenz über den Bereitstellungs-Workflow aller zu implementierender Arbeitselemente zu erlangen [7, Z. 89 ff.]. Auf diese Weise kann der CI/CD-Status eines Backlog-Items unmittelbar über die Planungs-Software eingesehen werden. Da die Wahl eines Pipeline-Tools i.d.R. nicht von der Unterstützung eines bestimmten Projektmanagement-Tools abhängig gemacht wird, erfolgt lediglich eine Untersuchung der generellen Integrationsfähigkeit von Planungs-Software [7, Z. 89 ff.]. In Kriterium K3 erfolgt die Evaluation der **Kosten** [18, Z. 43]. Mit diesem Entscheidungskriterium werden die durch die CI/CD-Pipelines verursachten *Total Cost of Owner Ship (TCO)* evaluiert. TCO beschreiben den Geldbetrag, welchen ein Un-

ternehmen während des gesamten Lebenszyklus einer Investition, also von Beschaffung bis zur vollständigen Entsorgung, aufbringen muss [5, S. 3]. Da für die zu untersuchenden Pipeline-Tools keine vergleichbaren Kostenmodelle verfügbar sind, werden die Kosten ausschließlich qualitativ beurteilt. Somit werden Aspekte wie Einrichtungs-, Betriebs- sowie Wartungsaufwendungen evaluiert. In Kriterium K4 wird die **Skalierbarkeit** der CI/CD-Pipelines analysiert [29, Z. 69 ff.]. Hierbei werden die Pipelines auf horizontale sowie vertikale Skalierbarkeit untersucht. Die horizontale Skalierbarkeit ermöglicht eine parallele Durchführung mehrerer Builds. Gerade bei einer hohen Anzahl gleichzeitiger Hauptzweigintegrationen birgt dies einen hohen Mehrwert. Die vertikale Skalierung bezieht sich auf die Erhöhung der Ressourcen einer Pipeline-Instanz. Laut Experte 1 kann die CI/CD-Pipeline so dynamisch an die sich ändernden Anforderungen angepasst werden [29, Z. 70 ff.]. In Kriterium K5 wird die **Performance** der Pipelines verglichen [18, Z. 36 ff.]. Dabei wird die zur Prozessierung des CI/CD-Workflows benötigte Zeit der zu untersuchenden Tools anhand eines für die Arbeit implementierten CEA-Prototyps evaluiert (s. 18). Im Rahmen dieser Gegenüberstellung wird eine Unterscheidung zwischen der Integration- bzw. Delivery-Zeit realisiert. Die Integration-Zeit bezeichnet den Zeitraum, welcher von der Einführung eines Feature-Branches bis zur vollständigen Konsolidierung in den Hauptzweig benötigt wird. Dabei werden für die Microservices dem CI-Prozess entsprechende Validierungen wie Unit- und Integration-Tests implementiert. Diese werden zur automatisierten Ausführung in die CI-Pipeline eingebunden. Die Delivery-Zeit beschreibt die Zeitspanne, welche von der Freigabe des Hauptzweigs bis zur Bereitstellung der Software auf die Cloud-Plattform benötigt wird. Dabei werden ebenfalls CD-typische Schritte, wie E2E-Tests und Code-Analysen implementiert und in die Pipelines integriert. In Kriterium K6 wird die **Flexibilität** der verschiedenen Pipelines evaluiert [29, Z. 65 ff.]. Eine bedeutende Dimension der Flexibilität ist die uneingeschränkte Konfigurierbarkeit der Pipelines. So sollte eine Pipeline laut Experte 1 etwa keinerlei Beschränkungen in Bezug auf Anzahl und Reihenfolge der im CI/CD-Workflow durchzuführenden Schritten besitzen [29, Z. 65 ff.]. Weiterhin wird evaluiert, ob die Pipelines einen modularen Aufbau unterstützen. Um die aus

einer Bereitstellungslandschaft mit einer Vielzahl an CI/CD-Pipelines resultierende Komplexität zu reduzieren, sollten Pipelines aus modularen wiederverwendbaren Komponenten zusammengesetzt werden können. Sofern eine neue Pipeline erforderlich ist, besteht die Möglichkeit diese ohne hohen Implementierungsaufwand aus den wiederverwendbaren CI/CD-Komponenten zu konstruieren. Ein weiterer, für die Flexibilität der Pipelines essenzieller Aspekt ist die Unterstützung von Plug-ins. Damit diesen ebenfalls nicht im Standard verfügbare Funktionalitäten in die Pipeline integriert werden können, besteht für die Entwicklungsabteilung die Möglichkeit flexibel auf Anforderungen aller Stakeholder zu reagieren. In Kriterium K7 wird der für die CI/CD-Pipelines bereitgestellte **Support** evaluiert. Im Hinblick auf den *Administrativen Support (K7.1)* wird geprüft, ob die Pipeline-Anbieter Unterstützung bei der Einrichtung, Konfiguration sowie Problembehebung der CI/CD-Tools bieten [18, Z. 44 ff.]. Dies ist insbesondere dann hilfreich, wenn der Umgang mit den Pipelines einen hohen Grad an Expertise erfordert. Des Weiteren wird evaluiert, ob Schulungen sowie Informationsmaterial verfügbar sind. Ein weiterer wesentlicher Aspekt ist die Verfügbarkeit von Updates. Durch kontinuierliche Updates kann sichergestellt werden, dass die Pipeline stets auf dem neusten Stand der Technik ist. Im Kontext des *Community-Supports (Kriterium K7.2)* wird geprüft, ob öffentliche Foren existieren, in welchen Anwender Fragen stellen und Probleme diskutieren können [18, Z. 45 ff.]. Die Qualität des Community-Supports hängt dabei i.d.R. von dem Kontributionsmaß innerhalb der Foren ab. Somit wird als Bewertungsgrundlage die Anzahl der zu einer CI/CD-Pipeline abgesetzten Posts verglichen. Als Referenzquelle dient hierbei das größte Entwicklerforum Stack-Overflow [0]. In dem Kriterium K8 wird die **Sicherheit** der CI/CD-Pipelines untersucht [29, Z. 75 ff.]. Um unerwünschte Zugriffe zu vermeiden, sollte die CI/CD-Pipeline ein Authentifizierungs- und Authentisierungskonzepte unterstützen. Besonders vorteilhaft ist dabei die Einbindung zentralisierter Drittanbieter, wie der SAP Identity Provider oder GitHub. Ein weiteres unter dem Kriterium der Sicherheit evaluierter Aspekt ist ebenfalls die Systemsicherheit. Dazu gehört neben dem Schutz der Systemintegrität ebenfalls die Ausfallsicherheit. In Kriterium K9 wird die **Benutzerfreundlichkeit** der CI/CD-

Pipelines untersucht. Hinsichtlich der *Installation und Wartung (Kriterium K9.1)* ist es dabei besonders vorteilhaft, wenn Installation und Wartung des CI/CD-Tools nicht selbst übernommen werden muss, sondern unmittelbar als Service bereitgestellt wird [29, Z. 65 ff.]. Auch der für die Implementierung und Konfiguration der Pipelines benötigte Aufwand sollte so gering wie möglich sein (*intuitive Bedienbarkeit*) [29, Z. 65 ff.]. Um die Abhängigkeit einer Abteilung von hochqualifizierten DevOps-Spezialisten zu verringern, kann es einen Mehrwert darstellen, wenn Pipelines nicht mittels Programmiersprachen, sondern über intuitive Benutzeroberfläche konfigurierbar sind.

4.2 Festlegung der Bewertungsmetriken

In diesem Abschnitt erfolgt die Festlegung der Bewertungsmetriken. Dabei ist eine Bewertung von null bis vier Punkte vorgesehen. Eine Bewertung mit vier Punkten wird vergeben, wenn eine CI/CD-Pipeline signifikant zur Zielerreichung eines Kriteriums beiträgt, während eine Bemessung mit null Punkten eine unterdurchschnittliche Leistung impliziert. Die Vergabe von null Punkten stellt sicher, dass ein Kriterium, falls die zu bewertende CI/CD-Pipeline keinen Mehrwert birgt, nicht zur Erhöhung der Gesamtbewertung beiträgt.

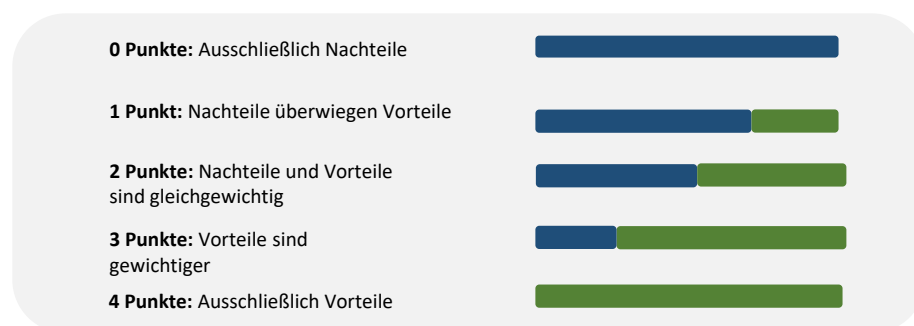


Abbildung 13: Qualitative Bewertungsmetrik für AHP. Eigene Darstellung.

Für qualitative, in dem Entscheidungs-Framework zu betrachtende Kriterien wird eine gewichtende Bewertung vorgenommen. Dabei erfolgt eine Abwägung der Vor- und Nachteile, welche sich aus der Nutzung einer bestimmten Pipeline ergeben. Auf diese

Weise kann bei der Bewertung ebenfalls argumentativ auf die in einer CEA vorliegenden Bedürfnisse eingegangen werden. Aufgrund des qualitativen Evaluations-Designs wird diese Metrik für *Funktionalität (K1)*, *Integrationsmöglichkeiten (K2)*, *Skalierbarkeit (K4)*, *Flexibilität (K5)*, *Administrativer Support (K7.1)*, *Sicherheit (K8)* und *Benutzerfreundlichkeit (K9)* angewendet. Da für die verschiedenen Pipelines divergierende Preismodelle festgelegt sind, kann für das Kriterium *Kosten (K3)* ebenfalls ausschließlich eine qualitative Erörterung abgewickelt werden. Für das quantitative Kriterium *Performance (K5)* wird eine divergierende Metrik definiert:

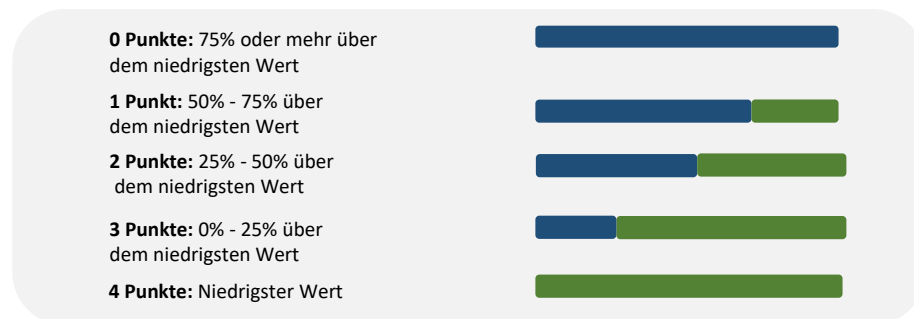


Abbildung 14: Quantitative Bewertungsmetrik für AHP. Eigene Darstellung.

Mithilfe dieser Metrik lassen sich die relativen Kosten der CI/CD-Pipelines vergleichend betrachten. Indem der höchste Wert als Bezugsgröße verwendet wird, ermöglicht sich ein Vergleich in Relation zur besten Entscheidungsalternative. Das vorliegende Evaluations-Design erweist sich dabei als besonders vorteilhaft, da dieses eine Bewertung ermöglicht, ohne vorab spezifische Referenzwerte festzulegen. In Kriterium K7.2 (*Community-Support*) ist ebenfalls eine quantitative Bewertung vorgesehen. Dabei wird die Anzahl der abgesetzten Blog-Posts zu einer CI/CD-Lösung evaluiert. Hierbei erfolgt eine inverse Bewertung der in Abb. 14 dargestellten Metrik. So werden vier Punkte für die höchste Blog-Post-Anzahl vergeben. Für die restlichen Bewertungen wird invers nach dem in Abb. 14 definierten Abstufungen benotet.

4.3 Ermittlung der Gewichtungsfaktoren

Zur Bestimmung einer optimalen, auf die Präferenzen der Entscheidungsträger ausgerichteten CI/CD-Pipeline wird eine Gewichtung der Bewertungskriterien vorge-

nommen. Die Gewichtung wird dabei von einem Expertengremium, bestehend aus fünf Mitarbeitenden der SAP, durchgeführt. Dieses umfasst einen Software-Architekten, einen Backend- sowie Frontend-Test-Entwickler, einen Fullstack-Entwickler und einen Product Manager (s. Anhang C.6). Dabei wird die in Kapitel 3.2 erläuterte Paarvergleichsgewichtung von jedem Experten zunächst eigenständig durchgeführt. Im Anschluss erfolgt die Berechnung des Mittelwertes aller Einzelgewichtungen. Mit diesem Vorgehen wird gewährleistet, dass bei der Gewichtung divergierende Anforderungen und Bedürfnisse aller an dem Bereitstellungsprozess von Software beteiligten Stakeholder adäquat berücksichtigt werden.










Kriterium	Lokale Gewichtung pro hundert	Globale Gewichtung pro hundert
 K1 Funktionalität	0,1728	0,1728
K1.1 Tests	0,3200	0,0548
K1.2 Code-Analysen	0,2080	0,0212
K1.3 Build	0,2240	0,0380
K1.4 Deploy	0,1200	0,0390
K1.5 Monitoring	0,1280	0,0198
 K2 Integrationsmöglichkeiten	0,1580	0,1580
K2.1 Integration in Repository	0,2889	0,0796
K2.2 Integration in Entwicklungsumgebung	0,5111	0,0444
K2.3 Integration in Planungs-Software	0,2000	0,0340
 K3 Kosten	0,0963	0,0963
 K4 Skalierbarkeit	0,1160	0,1160
 K5 Performance	0,0790	0,0790
K5.1 Integration-Zeit	0,7000	0,0574
K5.2 Delivery-Zeit	0,3000	0,0216
 K6 Flexibilität	0,1185	0,1185
 K7 Support	0,0815	0,0815
K7.1 Administrativer Support	0,3500	0,0290
K7.2 Community Support	0,6500	0,0525
 K8 Sicherheit	0,1259	0,1259
 K9 Benutzerfreundlichkeit	0,0519	0,0519
K9.1 Installation und Wartung	0,4000	0,0222
K9.2 Intuitive Bedienbarkeit	0,6000	0,0296

Abbildung 15: Gewichtungsfaktoren der AHP-Entscheidungskriterien. Eigene Darstellung.

In Abb. 15 werden die lokalen und globalen Durchschnittsgewichtungen der AHP-Entscheidungskriterien dargestellt (s. Abb. 15). Die Experten legen dabei auf unterschiedliche Aspekte Wert. So ist für den Software-Architekten die von den Pipelines bereitgestellte Funktionalität von hoher Bedeutung. Für den Experten besonderes ausschlaggebend ist dabei die Unterstützung verschiedener Test-Frameworks. Nach seiner Auffassung besteht die Möglichkeit, Tests manuell und unabhängig von einer Pipeline auszuführen. Allerdings gestaltet es sich dabei für Software-Architekten

sehr herausfordernd, die Einhaltung dieser Testrichtlinien durch die Entwickler angemessen zu überblicken [24, Z. 16]. Weiterhin ist für den Experten essenziell, dass Pipelines unterschiedliche Build-Tools unterstützen. So ist es bei einer CEA üblich, dass eine hohe Bandbreite verschiedener Programmier-Frameworks eingesetzt wird [24, Z. 11]. Da Pipeline-Systeme i.d.R. einmalig aufgesetzt werden, ist die Installation und Wartung für den Software-Architekten weniger wichtig. Sowohl für den Fullstack- als auch die Test-Entwickler stellt die Unterstützung automatisierter Tests einen signifikanten Aspekt dar [1, Z. 4]. Da die Entwickler ein möglichst zeitnahes Feedback auf Erweiterungen erhalten möchten, ist dabei insbesondere die Integration-Zeit essenziell [6, Z. 21]. Kosten stellen für die Entwickler hingegen eine geringe Relevanz dar, da diese während der operativen Tätigkeit kaum Berührungspunkte mit diesem Aspekt aufweisen. Eine höhere Bedeutung besitzt für die Entwickler hingegen der Community-Support [8, Z. 24]. Dieser kann als Hilfestellung zur Implementation der Pipelines verwendet werden. Für den Product Manager sind die Integrationsmöglichkeiten ein wichtiger Aspekt. Gemäß empirischer Erkenntnisse ist die Wahl eines CI/CD-Tools, insbesondere von der Kompatibilität mit dem Repository abhängig [17, Z. 3]. Weiterhin erachtet der Experte Sicherheit als einen essenziellen Aspekt. Da die Bereitstellung von Software zur Wertschöpfung beiträgt, können Unterbrechungen der CI/CD-Pipelines erhebliche Auswirkungen auf die Wettbewerbsfähigkeit besitzen. Überdies bieten CI/CD-Pipelines eine effektive Möglichkeit, um Schadsoftware in die Produktionssysteme einzuschleusen [17, Z. 6].

4.4 Bewertung der Entscheidungsalternativen

Das Erste zu bewertende Kriterium ist die **Funktionalität**. Sowohl für Jenkins als auch für Azure Pipelines wird die Programmbibliothek Project Piper verwendet. Mit dieser werden essenzielle, für die Bereitstellung von SAP-Technologien verwendete Pipeline-Funktionalitäten ausgeliefert. Daher erzielen beide CI/CD-Tools innerhalb der *Funktionalität* weitgehend ähnliche Ergebnisse. In Kriterium K1 (*Tests*) wird die Unterstützung von Unit-, Integration-, E2E-, sowie Regression-Tests evaluiert. Mit

Project Piper wird eine Test-Laufzeitumgebungen für Node zur Verfügung gestellt. Somit ist es möglich, Backend-Unit-Tests mittels Mocha und Jest auszuführen. Die Programmbibliothek unterstützt ebenfalls die für Frontend-Unit-Tests mittels Qunit und Frontend-Integration-Tests mittels OPA5 benötigte Laufzeitumgebung Karma. Darüber hinaus wird das für Backend-Integration-Tests benötigte Newmann-Tool bereitgestellt. Für E2E-Tests mittels WDI5 wird eine Webdriver-Laufzeitumgebung ausgeliefert [7, Z. 63 ff.]. Im Kontext der CEA stellt ebenfalls die Automatisierung von Regression-Tests einen essenziellen Faktor dar. Bei der Weiterentwicklung eines Microservice muss validiert werden, ob abhängige Dienste stets ordnungsgemäß funktionieren. Dafür wird mit Azure Pipelines und Jenkins ein Ressourcen-Trigger bereitgestellt [0]. Dieser gewährleistet ein automatisiertes Ausführen von Pipelines abhängiger Microservices. Dabei erfolgt die Veröffentlichung des neuen Artefakts erst, nachdem alle Tests validiert wurden. In SAP CI/CD können alle Tests, mit Ausnahme der Backend-Integration-Tests mittels Newmann und Regression-Tests ausgeführt werden. Dies stellt insbesondere für CEs einen erheblichen Nachteil dar. Diese sind aufgrund ihrer modularen IT-Architektur darauf angewiesen, dass einzelne Microservices reibungslos miteinander interagieren. Da ein implizites Validieren des Komponentenzusammenspiels ebenfalls über E2E-Tests abgewickelt wird, fällt dieser Nachteil jedoch weniger gewichtig aus. Aus diesem Grund wird eine Bewertung von drei Punkten für SAP CI/CD vergeben (Vorteile überwiegen Nachteilen). Für Azure Pipelines sowie Jenkins ist eine Bewertung mit vier Punkten vorgesehen (ausschließlich Vorteile). In Kriterium K1.2 erfolgt eine Validierung der durch die Pipeline bereitgestellten *Code-Analyse-Funktionalitäten*. Project Piper unterstützt statische Code-Analyse mittels Lint bzw. SonarQube, Sicherheitsüberprüfungen mittels Checkmarx und DASTER sowie Performance-Tests mittels JMeter [19, Z. 43 ff.]. Mit dem SAP CI/CD-Tool sind ausschließlich statische Code-Analysen mittels Lint bzw. SonarQube möglich [29, Z. 47 ff.]. Neben der fehlenden Unterstützung von Performance-Tests birgt insbesondere die Inkompatibilität von Sicherheits-Tools für CEA erhebliche Nachteile. Die CE-typische Verwendung von APIs zur Kommunikation zwischen einzelnen Microservices hat zur Folge, dass eine für unautorisiert-

te Zugriffe begünstigte Angriffsfläche entsteht. Obwohl mögliche Sicherheitsbedenken auch durch Security-Experten manuell behoben werden könnten, ist dies für die von CEs angestrebte dynamische Reaktionsfähigkeit hinderlich. Zudem wird die Abwicklung von *Static Application Security Testing (SAST)* mit Checkmarx bzw. *Dynamic Application Security Testing (DAST)* mit DASTER von der SAP als Produktqualitätsstandard vorgeschrieben [19, Z. 37 ff.]. Somit kann die SAP CI/CD-Pipeline nicht von Kunden verwendet werden, welche sich an dem Sicherheitsstandard der SAP orientieren. Aus diesem Grund ergibt sich eine Bewertung von einem Punkt für den SAP CI/CD-Service (Nachteile überwiegen) und vier Punkten für Jenkins bzw. Azure Pipelines (ausschließlich Vorteile). In Kriterium K1.3 wird die *Build-Funktionalität* der Pipelines evaluiert. Mit Project Piper wird dabei das für SAP UI5 sowie SAP CAP Node benötigte MTA-Build-Tool unterstützt. Weiterhin wird durch die Programmbibliothek ein Build-Tool für Docker-Container bereitgestellt [0]. Die kompilierten Anwendungsprogramme können mit Project Piper in einem Artefakt-Repository bereitgestellt werden. SAP CI/CD unterstützt kein Docker-Workflow sowie Repository-Artefakt [0]. Besonders gewichtig ist dabei die fehlende Unterstützung des Artefakt-Repositories. Artefakt-Repositories spielen für CEs eine essenzielle Rolle bei der Durchführung von Rollbacks. Bei dem Auftreten von Fehlern kann zu einer im Artefakt-Repository abgelegten früheren Version zurückgekehrt werden. Somit wird das Risiko von Ausfällen und Unterbrechungen im Geschäftsbetrieb minimiert. Da der SAP CI/CD-Service dennoch das für SAP CAP Node und SAP UI5 benötigte MTA-Build-Tool bereitstellt und somit den minimal erforderlichen Satz an benötigter Build-Funktionalität unterstützt, wird eine Bewertung von zwei Punkten veranschlagt (Nachteile und Vorteile gleichgewichtig). Für Azure Pipelines sowie Jenkins wird eine Bewertung von 4 Punkten vergeben (ausschließlich Vorteile). In Kriterium K1.4 erfolgt die Evaluierung der *Deploy- und Release-Funktionalitäten*. Dabei herrscht bei SAP CI/CD, Azure Pipelines sowie Jenkins Feature-Parität. Ein erheblicher Mehrwert besteht insbesondere darin, dass alle zu untersuchende CI-CD-Lösungen eine Bereitstellung in das SAP CTM unterstützen. Durch den Einsatz dieses Tools lässt sich die Bereitstellung von

Software innerhalb komplexer ERP-Systemlandschaften optimieren. Ein Unternehmen könnte etwa separate Systeme für das Entwickeln (*DEV*), Testen (*TEST*) sowie für die Produktionsumgebung (*PROD*) besitzen. Während die Verantwortung der Entwickler dabei auf die ordnungsgemäße Bereitstellung der Funktionalitäten in das DEV-System beschränkt ist, werden nachfolgende Schritte von dem Betriebsteam über das SAP CTM verwaltet. Über dieses System können dabei vielfältige Release-Konfigurationen, wie z.B. eine Zeitplan-gesteuerte Bereitstellung vorgenommen werden. Des Weiteren ermöglicht das SAP CTM die Definition von Abhängigkeiten verschiedener Services. Experte 2 merkt an, dass dies insbesondere für CEA von hoher Bedeutung ist [18, Z. 64]. Im Falle einer API-Änderung bestimmter Microservices, besteht die Möglichkeit, dass in konsumierenden Diensten entsprechend Fehler auftreten. Mittels des SAP CTMs kann dabei jedoch reguliert werden, dass die neue Version eines Microservices erst nach Anpassung der abhängigen Dienste in das Produktivsystem eingeführt wird (s. Abb. 16).

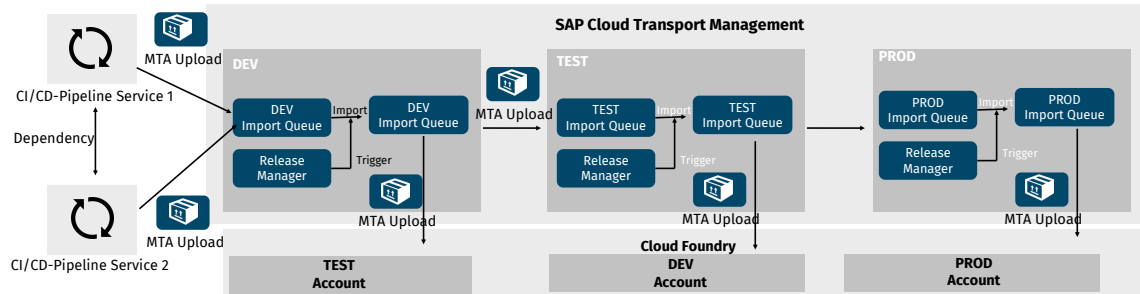


Abbildung 16: SAP Cloud Transportmanagement. In Anlehnung an Stevens [0].

Weiterhin wird von den Pipelines das Ausrollen einer neuen Anwendungsversion mittels Blue/Green-Deployment unterstützt [0]. Diese Strategie gewährleistet die notwendige Flexibilität und Agilität, welche es in CEs benötigt. Da im Blue-Green-Deployment neben der zu installierenden neuen Version ebenfalls die stabile Anwendung betrieben wird, können mit dieser Bereitstellungsstrategie Unterbrechungszeiten vermieden werden. Dies spielt insbesondere im Kontext von Composable-ERP-Systemen, in welchem etwa kritische Finanzprozesse abgewickelt werden, ei-

ne bedeutende Rolle. Während alle CI/CD-Tools eine Cloud-Froundry-, CTM- sowie Blue-Green-Bereitstellung unterstützen, ist Canary- sowie Shadow-Deployment nicht möglich. Da diese jedoch ausschließlich ergänzende Zusatzfunktionalitäten und keine essenziell benötigten Werkzeuge darstellen, wird eine Bewertung von 3 Punkten vergeben (Vorteile überwiegen). In Kriterium K1.5 wird die *Monitoring-Funktionalität* der Pipelines untersucht. Mit Project Piper können im CI/CD-Prozess generierte Logs unmittelbar an das Monitoring-Dashboard Splunk übermittelt werden [18, Z. 72 ff.]. Open-Soruce-Tools wie das Kibana-Dashboard lassen sich dabei ebenfalls mit Jenkins bzw. Azure Pipelines verknüpfen. Während Kibana unmittelbar als Azure-SaaS-Tool verfügbar ist, benötigt das Aufsetzen auf Jenkins einen deutlich höheren Aufwand (s. Abb. 17).

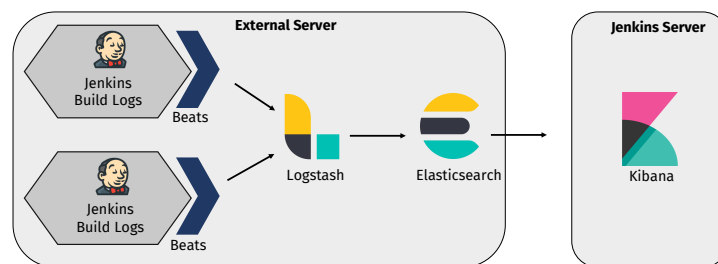


Abbildung 17: Pipeline-Monitoring mit Kibana und Jenkins. In Anlehnung an Atta [0]

So müssen auf der Jenkins-Plattform Tools zum Versenden (Beats), Transformieren (Logstash) bzw. zum Speichern (Elasticsearch) der Pipeline-Logs manuell aufgesetzt werden. Um die Pipeline-Metriken letztlich auf dem Dashboard zu visualisieren, müssen die Logs über APIs an eine extern gehostete Kibana-Instanz übermittelt werden [0]. Der SAP CI/CD-Service unterstützt ausschließlich eine Protokollierung der Build-Logs einzelner Pipelines. Für Experte 2 stellt die Inkompatibilität von Monitoring-Dashboards insbesondere in einer Microservice-Architektur einen erheblichen Nachteil dar. So sind die zentralen Dashboards bei aus komplexen Diensten bestehenden Systemarchitekturen häufig die einzige Möglichkeit CI/CD-Prozesse nachhaltig zu überwachen [18, Z. 39]. Während für Azure Pipelines vier Punkte vergeben werden (ausschließlich Vorteile), erhält Jenkins aufgrund des hohen Ein-

richtungsaufwands für Kibana-Dashboard eine Bewertung von drei Punkten (Vorteile überwiegen). Für die SAP CI/CD-Pipeline wird aufgrund der fehlenden Unterstützung von Monitoring-Dashboards ausschließlich ein Punkt vergeben (Nachteile überwiegen). In Kriterium K2 werden die **Integrationsmöglichkeiten** der Pipelines evaluiert. Alle drei Pipelines unterstützen dabei eine Integration der Repositories GitHub, BitBucket und GitLab (Kriterium K2.1). Für Jenkins und SAP CI/CD müssen dafür jedoch manuell in den entsprechenden Repositories Webhooks aufgesetzt werden. Ein Webhook ist eine API, welche Anfragen zum Auslösen eines CI/CD-Workflows an die Pipeline übermittelt. Diese Webhook-Anfragen können bei einer Vielzahl von Events, einschließlich dem *Pushen* von Codeänderungen oder dem Eröffnen von *Pull-Requests*, ausgelöst werden. SAP CI/CD unterstützt dabei keine Pull-Request-Webhooks. Dies birgt im Entwicklungsprozess erhebliche Nachteile. Eine Pull-Request-Pipeline gewährleistet, dass Entwicklungen eines Feature-Branche erst nach erfolgreicher Abwicklung aller Tests in den Hauptzweig integriert werden [29, Z. 27 ff.]. Ohne diese Funktionalität muss die Pipeline bei Erstellung eines Pull-Requests stets manuell gestartet und überprüft werden, was zur Beeinträchtigung der Zusammenarbeit in Teams führt. Für Azure Pipelines wird eine Bewertung von vier Punkten vergeben (ausschließlich Vorteile). Jenkins erhält aufgrund der Notwendigkeit einer manuellen Webhooks-Konfiguration drei Punkte (Vorteile überwiegen). Da SAP CI/CD darüber hinaus keine Pull-Request-Pipelines unterstützt und diese laut Experte 1 in einigen Entwicklungsabteilungen gängige Praxis darstellen, wird eine Bewertung von zwei Punkten vergeben (Vorteile und Nachteile gleichgewichtig) [29, Z. 27 ff.]. In Jenkins sowie Azure Pipelines lassen sich die Entwicklungsumgebungen Microsoft Visual Studio Code sowie Eclipse integrieren (*Integrationsmöglichkeiten von Repositories (Kriterium K2.2)*). Die SAP CI/CD-Pipeline unterstützt keine der zu evaluierenden Entwicklungsumgebungen [29, Z. 91 ff.]. Somit ist es Entwicklern nicht möglich, die CI-Pipeline unmittelbar aus der Entwicklungsumgebung zu starten. Da sowohl für Azure Pipelines als auch Jenkins ausschließlich zwei der drei zu evaluierenden Entwicklungsumgebung unterstützt werden, wird eine Bewertung von drei Punkten vergeben (Vorteile überwiegen Nach-

teilen). SAP CI/CD wird aufgrund der fehlenden Integrationsmöglichkeiten mit null Punkten bewertet (ausschließlich Nachteile). Sowohl mit Azure Pipelines als auch Jenkins lassen sich Planungs-Tools wie Jira integrieren (*Integrationsmöglichkeiten von Planungs-Software (Kriterium K2.2)*). Für den SAP BTP-Service besteht diese Integrationsmöglichkeit nicht [7, Z. 98 ff.]. Demnach ist es Projektmanagern nicht möglich, den Build-Status verschiedener Backlog-Items zu begutachten. Aus diesem Grund ergibt sich eine Bewertung von vier Punkten für Azure Pipelines sowie Jenkins bzw. null Punkte für den SAP BTP-Service. In Kriterium K3 werden die **Kosten** der CI/CD-Pipelines evaluiert. Für SAP CI/CD werden Gebühren von einem Euro pro Build-Stunde fällig [29, Z. 103 ff.]. Azure Pipelines berechnet hingegen unabhängig von der Nutzungszeit 40 Euro pro Pipeline [0]. In den genannten Preisen werden Kosten für die IT-Infrastruktur, Installation, Wartung sowie Support berücksichtigt. Zwar fallen für die Jenkins-Pipeline keine Lizenzgebühren an, jedoch müssen im Gegensatz zu den SaaS-Tools weitere Kostenpositionen beachtet werden. Da Jenkins in einem On-Premise-Modell betrieben wird, müssen hierbei zunächst Investitionskosten für Hardware berücksichtigt werden. Weiterhin fallen für den On-Premise-Server laufende Betriebskosten, wie Ausgaben für Energie, Reparaturleistungen und Personal zur Wartung der IT-Infrastruktur an. Untersuchungsergebnisse des IT-Beratungshauses ExecuTech zeigen, dass On-Premise-Systeme aufgrund hoher Investitionskosten insbesondere bei einer kurzfristigen Betrachtung teurer sind [0]. Eine langfristige Perspektive führt jedoch zu einem anderen Ergebnis. Da die laufenden Betriebskosten für On-Premise-Systeme häufig geringer als die SaaS-Gebühren sind, können diese auf lange Sicht kosteneffektiver werden. CEs sind jedoch darauf ausgerichtet, flexibel und agil zu agieren, um auf sich ändernde Geschäftsanforderungen reagieren zu können. Somit müssen diese in der Lage sein, Systeme schnell auf- und abzubauen. Dies würde im On-Premise-Modell kontinuierliche Investitionskosten verursachen, wobei SaaS ebenfalls auf längere Sicht die günstigere Alternative bleibt. Aus diesem Grund wird für die SaaS-Systeme Azure Pipelines und SAP CI/CD eine Bewertung von drei Punkten bzw. für Jenkins einen Punkt vergeben (Nachteile überwiegen Vorteilen). In Kriterium K4 wird die **Ska-**

lierbarkeit der Tools evaluiert. Mit Azure Pipelines lässt sich die CI/CD-Pipeline eines Microservices horizontal skalieren. Um parallele Builds auszuführen, werden hierbei mehrere *Microsoft-hosted Agents* aktiviert. Ein Microsoft-hosted Agent ist eine von Azure verwaltete virtuelle Maschine, welche eine vorkonfigurierte Build- und Testumgebung bereitstellt [0]. Eine vertikale Skalierung kann dabei durch das Zuweisen zusätzlicher Ressourcen zu einem Microsoft-hosted Agent erreicht werden. Während eine horizontale Skalierung ebenfalls über das Hinzufügen zusätzlicher Agents ermöglicht wird, ist eine vertikale Skalierung aufgrund architektonischer Limitationen bei Jenkins nur begrenzt möglich [0]. So müssen in das bestehende System zusätzliche Hardware-Ressourcen integriert werden. Das bedeutet, dass die vertikale Skalierbarkeit auf die physischen Einschränkungen eines Servers begrenzt ist. Während horizontale Skalierung von SAP CI/CD unterstützt wird, ist eine vertikale Skalierung nicht realisierbar. Die vertikale Skalierbarkeit von CI/CD-Pipelines spielt dabei insbesondere in einem volatilen Geschäftsumfeld eine signifikante Rolle. Als eines der größten CEs stellt der Streaminganbieter Netflix über 700 Microservices bereit. Angesichts der Notwendigkeit, für jeden Microservice mindestens eine CI/CD-Pipeline bereitzustellen, würde ein nicht skalierbares Pipeline-System eine umfangreiche IT-Infrastruktur erfordern. Netflix setzt aus diesem Grund auf eine vertikal-skalierbare Container-Orchestrierungstechnologie [0]. Mit dieser werden zur Laufzeit Pipelines in Docker-Container bereitgestellt, welche je nach Bedarf auf- oder abgebaut bzw. skaliert werden können. So wird Azure Pipelines mit vier Punkten (ausschließlich Vorteile) bzw. Jenkins und SAP CI/CD mit einem Punkt bewertet (Nachteile überwiegen Vorteilen). In Kriterium K5 wird die **Performance** der Pipelines evaluiert. Zur Durchführung dieser Analyse wird ein speziell für die Arbeit entwickelter Prototyp einer CEA verwendet. Dieser besteht aus drei Microservices, welche jeweils ein Frontend (SAP UI5), eine Service-API (SAP CAP Node) sowie eine Datenbank (SAP HANA) besitzen (s. Abb. 18). Des Weiteren ist mit den zu untersuchenden Tools für jeden Microservice eine eigene CI- sowie CD-Pipeline implementiert. Die CI-Pipeline besteht hierbei aus einem MTA-Build, statischen Lint-Code-Analysen, Backend-Unit-Tests, Frontend-Unit- sowie -Integration-Tests. Die

CD-Pipeline erweitert den CI-Prozess um Sonar-Qube-Code-Analysen, End-To-End-Tests sowie einem Schritt zur Bereitstellung der Anwendung auf der SAP BTP. Im Vergleich zu den anderen zu evaluierenden Tools weist der SAP CI/CD den geringsten Funktionsumfang auf. Um einen besseren Vergleich zu gewährleisten, ist der Aufbau der anderen Pipelines strukturell an den Funktionsumfang dieses Tools angepasst. So sind etwa für keine der Pipelines Integration-Tests, sowie Security-Scans implementiert.

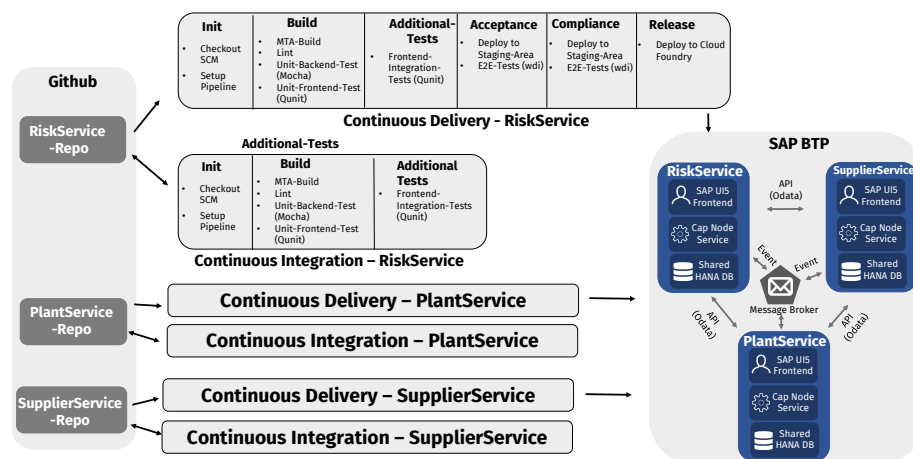


Abbildung 18: CEA-Szenario für Performance-Tests. Eigene Darstellung.

Für jedes zu untersuchende CI/CD-Tool wird dabei die Durchlaufzeit der Pipelines aller drei Microservices in Sekunden gemessen. Anschließend werden die Einzelmessungen der Services aggregiert, um für jedes zu evaluierende CI/CD-Tool eine Integration- sowie Delivery-Zeit zu erhalten (s. Anhang 21).



		CI-Pipeline 				CD-Pipeline 						
		Init	Build	Additional Tests	Σ	Init	Build	Additional Tests	Acceptance	Compliance	Release	Σ
Azure Pipelines	RiskService											
	SupplierService											
	PlantService											
	Σ											
Jenkins	RiskService	6	253	128	387	5	245	136	468	111	179	1144
	SupplierService	6	243	115	364	5	238	125	455	106	167	1096
	PlantService	6	247	130	383	6	246	134	493	113	178	1169
	Σ				1134							3409
SAP CI/CD	RiskService	48	249	115	412							
	SupplierService	48	247	119	414							
	PlantService	47	251	119	417							
	Σ				1243							

Tabelle 1: Integration- und Delivery-Zeit in Sekunden. Eigene Darstellung.

In Kriterium K6 wird die **Flexibilität** der Services evaluiert. Sowohl mit Azure als auch Jenkins lassen sich die mit der Programmbibliothek Project Piper bereitgestellten CI/CD-Schritte (z.B. Build, Test etc.) frei nach Bedarf kombinieren und konfigurieren. SAP CI/CD besitzt dabei maßgebliche Einschränkungen. Hierbei werden Anzahl, Reihenfolge sowie Konfiguration der Schritte von dem Service vorgeschrieben. Somit ist keine flexible Implementierung der Pipelines möglich. Laut Experte 5, stellt Technologieoffenheit einen fundamentalen Wert in einer CEA dar [24, Z. 10]. So sollte die CI/CD-Pipeline verschiedene Programmier-Frameworks, Test-Tools und Cloud-Plattformen unterstützen. Damit werden bei Jenkins mehr als 1500 Plug-ins bereitgestellt, womit Funktionalitäten, welche über den Standard hinausgehen, integriert werden können [0]. Während mit Azure Pipelines 1400 Plug-ins bereitgestellt werden, besteht diese Möglichkeit bei SAP CI/CD nicht [0]. Weiterhin sollten die Pipelines, um schnell neue Microservices bereitzustellen, einen flexiblen modularen Aufbau ermöglichen. Für Azure Pipelines und Jenkins wird durch die Verwendung von Project Piper das Shared-Library-Konzept realisiert (s. Abb. 19). Damit ist es möglich vorimplementierte Funktionalitäten, wie z.B. standardisierte Build, Deploy oder Test-Schritte in einer gemeinsamen Bibliothek zu konsolidieren. Eine weitere Möglichkeit, um die Komplexität der Bereitstellung zu reduzieren, besteht in der Wiederverwendung ganzer Pipelines. Dieses Konzept wird verwendet, wenn sämtliche Microservices einen homogenen Bereitstellungszyklus durchlaufen. Durch die Wiederverwendung ganzer Pipelines kann somit sichergestellt werden,

dass diese auf analoge Weise kompiliert, verpackt und bereitgestellt werden. Bei Jenkins und Azure lässt sich dies durch die Einbindung mehrerer Repositories in eine Pipeline realisieren. Bei SAP CI/CD wird ebenfalls ein Shared-Library-Konzept unterstützt. So werden die templatebasierten Schritte als standardisierte Elemente einer CI/CD-Pipeline ausgeliefert. Die Wiederverwendung ganzer Pipelines für unterschiedliche Microservices ist hingegen nicht möglich.

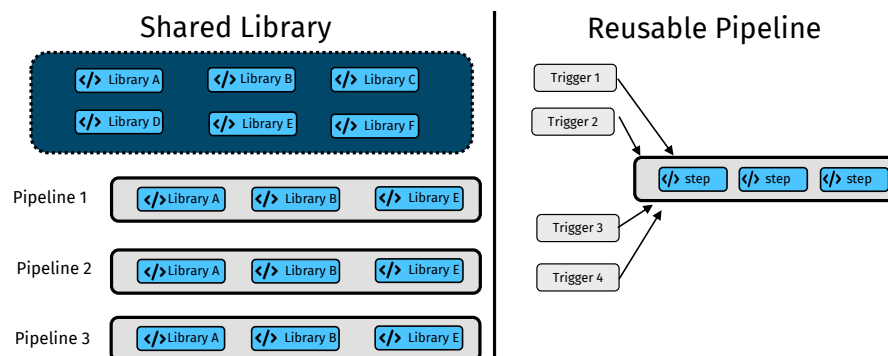


Abbildung 19: Modularer Aufbau einer CI/CD-Pipeline. In Anlehnung an Codefresh [0].

Während der SAP CI/CD-Service das Shared-Library-Konzept unterstützt, ist eine flexible Implementierung der Pipelines, die Einbindung von Plug-ins, sowie die Wiederverwendung von CI/CD-Pipelines nicht möglich. Deshalb wird für den SAP CI/CD-Service eine Bewertung von einem Punkt vergeben (Nachteile überwiegen Vorteilen). Da Jenkins und Azure Pipelines bei den zu untersuchenden Entscheidungskriterien nur Vorteile bieten, werden jeweils vier Punkte vergeben. In Kriterium K8 wird der **Support** der CI/CD-Tools evaluiert. Azure Pipelines stellt einen umfangreichen *administrativen Support* bereit (Kriterium K7.1). So besteht für Kunden die Möglichkeit einen Premium-Support in Anspruch zu nehmen [0]. Dabei handelt es sich um eine kostenpflichtige Leistung, welche den Kunden unmittelbaren Zugang zu technischem Support und Beratung gewährt. Die Unterstützung beinhaltet Installation bzw. Konfiguration von Pipelines, Integration einer vorhandenen CI/CD-Toolchain (Tests, Repositories etc.) einschließlich der Optimierung der Pipeline-Performance. Zudem wird über das Azure-Support-Center ein kostenfreies Support-Ticket-System für Nicht-Premium-Nutzer angeboten. Diese Kanäle können

verwendet werden, um allgemeine technische Probleme des CI/CD-Services zu melden. Weiterhin wird von Azure eine umfangreiche Dokumentation sowie Schulungsmaterialien, einschließlich Online-Tutorials, Videos und Webinare bereitgestellt. Der SAP CI/CD-Service stellt über den Drittanbieter Service Now ebenfalls ein kostenfreies Ticket-System zur Verfügung. Für spezifische Fragestellungen und individuelle Unterstützung besteht für Kunden die Möglichkeit, sich an die technische Beratung der SAP wenden [29, Z. 109 ff.]. Wie auch bei Azure Pipelines stehen für SAP CI/CD Schulungs- und Dokumentationsunterlagen zur Verfügung. Für Jenkins sind diese Informationsmaterialien ebenfalls vorhanden, jedoch ist der administrative Support aufgrund des Open-Source-Charakters im Vergleich zu den anderen Lösungen begrenzt. Dies ist insbesondere auf das Fehlen eines unmittelbaren Supports durch einen Service-Anbieter zurückzuführen. Ebenso spiegelt sich dieser Aspekt in der Verfügbarkeit von Updates wider. Viele der Funktionalitäten von Jenkins werden über Open-Source-Plug-ins bereitgestellt. Experte 6 merkt an, dass hierbei stets das Risiko besteht, dass die Wartung einzelner Plug-ins in Zukunft vernachlässigt werden könnte [8, Z.30]. Aus den aufgeführten Aspekten ergibt sich eine Bewertung von vier Punkten für Azure Pipelines sowie dem SAP CI/CD-Service (ausschließlich Vorteile). Für Jenkins wird eine Bewertung von einem Punkt vergeben (Nachteile überwiegen). Aufgrund des Open-Source-Charakters ist für die Jenkins-Pipeline ein umfassender *Community-Support* vorhanden (Kriterium K7.2). Jenkins ist dabei auf zahlreichen hochfrequentierten Foren vertreten. So wurden auf Stack-Overflow bereits 112000 Posts zur Jenkins-Pipeline veröffentlicht [0]. Dadurch wird den Anwendern Zugang zu einer breiten Palette an Ressourcen ermöglicht. Experte 4 merkt an, dass insbesondere die Gamifizierung dieser Plattformen zu einer schnellen und effektiven Unterstützung durch Spezialisten beiträgt. Auch Azure Pipelines sowie der SAP CI/CD sind ebenfalls auf öffentlichen Community-Foren vertreten, jedoch ist der dort publizierte Inhalt im Vergleich zu Jenkins signifikant geringer. Das liegt an der Tatsache, dass die genannten Pipelines als SaaS-Lösungen konzipiert sind und somit weniger Raum für fragebedürftige Anpassungen bestehen. So sind für Azure Pipelines 25000 Post und für SAP CI/CD lediglich 26 Einträge

auf Stack-Overflow veröffentlicht [0][0]. Gemäß der in Kapitel 4.2 festgelegten Metrik wird eine Bewertung von vier Punkten für Jenkins (höchste Blog-Post-Anzahl) bzw. von einem Punkt für Azure Pipelines und SAP CI/CD vergeben (0 Prozent bis 25 Prozent unter dem höchsten Wert). In Kriterium K8 wird die **Sicherheit** der CI/CD-Tools evaluiert. Jenkins bietet flexible Möglichkeiten zur Authentifizierung. Neben einer integrierten Benutzerverwaltung lassen sich über Jenkins ebenfalls Drittanbieterdienste wie Github, Google, BitBucket oder SAP-Single-Sign-On (SAP-SSO) einbinden [0]. Des Weiteren ist bei Jenkins die Implementierung eines Rollenkonzeptes möglich. Dabei können Rollen und Berechtigungen erstellt und bestimmten Benutzer zugewiesen werden. Damit wird sichergestellt, dass kritische Konfigurationen ausschließlich von Spezialisten vorgenommen werden [0]. Azure Pipelines und SAP CI/CD unterstützen ebenfalls Authentifizierungsmöglichkeiten. Die Implementierung eines Rollenkonzeptes kann im SAP CI/CD-Service jedoch nicht vorgenommen werden. Aufgrund der Integration eines Plug-in-Ökosystems erhöht sich im Kontext der Systemsicherheit für Jenkins das Risiko unbefugter Zugriffe. Da in der Jenkins-Community jeder Entwickler Plug-ins veröffentlichen kann, besteht die Möglichkeit, dass Sicherheitsrisiken mit diesen Erweiterungen einhergehen [24, Z. 39]. Bei Azure Pipelines ist die Einbindung von Plug-ins ebenfalls möglich. Da hierbei jedoch ausschließlich validierte auf dem Azure Marktplatz bereitgestellte Plug-ins integrierbar sind, fällt das korrespondierende Sicherheitsrisiko deutlich geringer aus. IT-Sicherheit gehört zum Kerngeschäft von Cloud-Providern wie Microsoft oder SAP gehört. Aus diesem Grund verfügen diese über erfahrenes Sicherheitspersonal. Somit lässt sich schlussfolgern, dass sowohl Azure Pipelines als auch SAP CI/CD ein hohes Niveau an Systemsicherheit bieten. Da Jenkins im On-Premise-Modell bereitgestellt wird, benötigt ein Unternehmen eigenes Sicherheitspersonal. Insbesondere in kleineren Organisationen kann es aufgrund finanzieller Einschränkungen jedoch vorkommen, dass diese kein hochqualifiziertes Sicherheitspersonal bereitstellen können. Jedoch bietet sich im On-Premise der Vorteil, dass Unternehmen vollständige Kontrolle über die Systemkonfiguration behalten. So ist diesem möglich, gezielte Maßnahmen zur Verbesserung der Sicherheit zu ergreifen.

Im Gegensatz dazu hängt die Sicherheit von SAP CI/CD bzw. Azure Pipelines in erheblichem Maße von den Cloud-Anbietern ab. Aufgrund der situativen Gegebenheiten werden Vor- und Nachteile der On-Premise- bzw. Cloud-Bereitstellung als gleichwertig betrachtet. Daraus resultiert sowohl für Azure Pipelines als auch für den SAP CI/CD-Service eine Bewertung von zwei Punkten. Aufgrund der zusätzlichen Sicherheitsrisiken im Kontext des Plug-in-Ökosystems wird für Jenkins eine Bewertung von einem Punkt vergeben (Nachteile überwiegen Vorteilen). In Kriterium K9 wird die **Benutzerfreundlichkeit** der Pipelines evaluiert. Da Azure Pipelines und SAP CI/CD als SaaS-Lösung vertrieben werden, bieten diese hinsichtlich der *Installation und Wartung (Kriterium K9.2)* einige Vorteile. Bei cloudbasierten Anwendungen entfällt die Notwendigkeit CI/CD-Systeme zu installieren und einzurichten. Auch die Wartung der IT-Infrastruktur liegt in Verantwortung der Cloud-Anbieter. Dadurch werden IT-Spezialisten zeitlich entlastet, wodurch Fachpersonal verstärkt auf die Kernkompetenzen des Unternehmens fokussiert werden können. Da Jenkins in einer On-Premise-Umgebung betrieben wird, obliegt die Verantwortung der Installation und Wartung den Unternehmen selbst. Daraus resultiert eine Bewertung von vier Punkten für Azure Pipelines und SAP CI/CD (ausschließlich Vorteile) bzw. von null Punkten für Jenkins (ausschließlich Nachteile). In Kriterium K9.2 wird die *intuitive Bedienbarkeit* der Tools evaluiert. Azure Pipelines und SAP CI/CD bieten eine webbasierte Benutzeroberfläche, über welche Build-Prozesse, Systemkonfigurationen und Pipelines eingesehen und angepasst werden können. Die Implementierung der Pipeline erfolgt dabei jedoch über Code. Während für Azure Pipelines eine YAML-basierte Syntax verwendet wird, werden CI/CD-Pipelines in Jenkins mit Groovy implementiert. Unter Verwendung von Project Piper kann eine Pipeline jedoch mit minimalem Aufwand an Code erstellt werden. Im Gegensatz dazu benötigt der SAP CI/CD-Service keine manuelle Implementierung über Code. So können Anwender mithilfe einer webbasierten Benutzeroberfläche die gesamte Implementierung einer Pipeline konfigurieren. Dies ist dabei insbesondere für CEs von wesentlicher Bedeutung. Bei der Bereitstellung neuer Microservices ist ggf. die Implementierung einer neuen Pipeline erforderlich. Eine intuitive Bedienbarkeit bietet dabei erhebliche Vor-

teile, um eine schnelle Erstellung von Pipelines zu gewährleisten. Aus diesem Grund erhält SAP CI/CD eine Bewertung von vier Punkten (ausschließlich Vorteile). Obwohl Project Piper grundsätzlich zur Vereinfachung der Implementierung beitragen kann, benötigt es einer aufwendigen Programmierung von Anforderungen, welche über den Bibliotheksstandard hinausgehen. Aus diesem Grund wird für Azure Pipelines sowie dem SAP CI/CD-Service eine Bewertung von einem Punkt vergeben (Nachteile überwiegen Vorteilen).










	B: Bewertung gB: gewichtete Bewertung	Globale Gewichtung pro hundert	Azure Pipelines		Jenkins		SAP CI/CD	
			B	gB	B	gB	B	gB
 K1 Funktionalität		0,1728	-	-	-	-	-	-
K1.1 Tests		0,0548	4	0,2192	4	0,2192	3	0,1644
K1.2 Code-Analysen		0,0212	4	0,0848	4	0,0848	1	0,0212
K1.3 Build		0,0380	4	0,1520	4	0,1520	2	0,0760
K1.4 Deploy		0,0390	3	0,1770	3	0,1770	3	0,1770
K1.5 Monitoring		0,0198	4	0,0792	3	0,0594	1	0,0198
 K2 Integrationsmöglichkeiten		0,1580	-	-	-	-	-	-
K2.1 Integration in Repository		0,0796	4	0,3076	3	0,2307	2	0,1538
K2.2 Integration in Entwicklungsumgebung		0,0444	3	0,1332	3	0,1332	0	0,0000
K2.3 Integration in Planungs-Software		0,0340	4	0,1360	4	0,1360	0	0,0000
 K3 Kosten		0,0963	3	0,2889	1	0,0963	3	0,2889
 K4 Skalierbarkeit		0,1160	4	0,4640	1	0,1160	1	0,1160
 K5 Performance		0,0790	-	-	-	-	-	-
K5.1 Integration-Zeit		0,0574						
K5.2 Delivery-Zeit		0,0216						
 K6 Flexibilität		0,1185	4	0,4740	4	0,4740	1	0,1185
 K7 Support		0,0815	-	-	-	-	-	-
K7.1 Administrativer Support		0,0290	4	0,1160	1	0,0290	4	0,1160
K7.2 Community Support		0,0525	1	0,0525	4	0,2100	1	0,0525
 K8 Sicherheit		0,1259	2	0,2518	1	0,1259	2	0,2518
 K9 Benutzerfreundlichkeit		0,0519	-	-	-	-	-	-
K9.1 Installation und Wartung		0,0222	4	0,0888	0	0,0000	4	0,0888
K9.2 Intuitive Bedienbarkeit		0,0296	1	0,0296	1	0,0296	4	0,1184

Tabelle 2: Ergebnistabelle zum AHP. Eigene Darstellung.

Unter Berücksichtigung der zu untersuchenden Fragestellung ergibt sich eine Gesamtbewertung von x für Azure Pipelines, x für Jenkins und x für SAP CI/CD (s. Tab. 2).

5 Entwicklung einer ganzheitlichen Bereitstellungsstrategie

6 Schlussbetrachtung

6.1 Fazit und kritische Reflexion

6.2 Ausblick auf zukünftige Entwicklungen

Literatur

Print-Quellen

- [1] Backend Test Developer SAP DTS Integration, Hrsg. *Expertengewichtung 4*. 29.03.2023.
- [2] Matthias Biehl. *API architecture: The big picture for building APIs*. API-university series. API-University Press, 2015. ISBN: 9781508676645.
- [3] Ralf Bruns und Jürgen Dunkel. *Event-Driven Architecture: Softwarearchitektur für ereignisgesteuerte Geschäftsprozesse*. Xpert.press. Berlin, Heidelberg: Springer-Verlag, 2010. ISBN: 9783642024399. DOI: 10.1007/978-3-642-02439-9.
- [4] Rong N. Chang u. a. „Realizing A Composable Enterprise Microservices Fabric with AI-Accelerated Material Discovery API Services“. In: *2020 IEEE 13th International Conference - 10/23/2020*, S. 313–320. DOI: 10.1109/CLOUD49709.2020.00051.
- [5] Lisa Ellram. „Total Cost of Ownership: Elements and Implementation“. In: *International Journal of Purchasing and Materials Management* 29.3 (1993), S. 2–11. ISSN: 10556001. DOI: 10.1111/j.1745-493X.1993.tb00013.x.
- [6] Frontend Test Developer SAP Hyperspace Adoption & Onboarding, Hrsg. *Expertengewichtung 3*. 22.03.2023.
- [7] Frontend Test Developer SAP Hyperspace Adoption & Onboarding, Hrsg. *Experteninterview 4*. 22.03.2023.
- [8] Full-Stack-Entwickler SAP DTS Integration, Hrsg. *Expertengewichtung 2*. 21.03.2023.
- [9] Kathleen Gerson und Sarah Damaske. *The science and art of interviewing*. New York, NY: Oxford University Press, 2021. ISBN: 9780199324293. DOI: 10.1093/oso/9780199324286.001.0001.
- [10] Joachim Goll und Daniel Hommel. *Mit Scrum zum gewünschten System*. Wiesbaden: Springer Vieweg, 2015. ISBN: 9783658107208. DOI: 10.1007/978-3-658-10721-5.

- [11] Jürgen Halstenberg. *DevOps: Ein Überblick*. Essentials Ser. Wiesbaden: Springer Fachmedien Wiesbaden GmbH, 2020. ISBN: 9783658314057. URL: <https://ebookcentral.proquest.com/lib/kxp/detail.action?docID=6380828>.
- [12] Brian Hambling und Brian, Hrsg. *Software testing: An ISTQB-BCS certified tester foundation guide*. 3rd ed. London, England: BCS Learning & Development Limited, 2015. ISBN: 9781780173016. URL: <https://learning.oreilly.com/library/view/-/9781780172996/?ar>.
- [13] Achim Hildebrandt u. a. *Methodologie, Methoden, Forschungsdesign: Ein Lehrbuch für fortgeschrittene Studierende der Politikwissenschaft*. Wiesbaden: Springer Fachmedien Wiesbaden, 2015. ISBN: 978-3-531-18256-8. DOI: 10.1007/978-3-531-18993-2.
- [14] Mohamed Labouardy. *Pipeline as code: Continuous delivery with Jenkins, Kubernetes, and Terraform*. Shelter Island, New York: Manning Publications Co, 2021. ISBN: 9781638350378. URL: <https://ebookcentral.proquest.com/lib/kxp/detail.action?docID=6785307>.
- [15] Jon Loeliger und Matthew McCullough. *Version control with git: Powerful tools and techniques for collaborative software development*. 2nd ed. Sebastopol, CA.: O'Reilly, 2012. ISBN: 9781449345051.
- [16] Dieter Masak. *Digitale Ökosysteme: Serviceorientierung bei dynamisch vernetzten Unternehmen*. Xpert.press. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. ISBN: 978-3-540-79129-4. DOI: 10.1007/978-3-540-79130-0.
- [17] Product Management CLM, Hrsg. *Expertengewichtung 5*. 29.03.2023.
- [18] Product Manager SAP Hyperspace CI/CD, Hrsg. *Experteninterview 2*. 24.03.2023.
- [19] Product Manager SAP Hyperspace Security Tools, Hrsg. *Experteninterview 3*. 22.03.2023.
- [20] Stefan Reinheimer. *Cloud Computing: Die Infrastruktur der Digitalisierung*. Edition HMD. Wiesbaden, Germany und Ann Arbor: Springer Vieweg und

- ProQuest EbookCentral, 2018. ISBN: 978-3-658-20966-7. DOI: 10.1007/978-3-658-20967-4.
- [21] Thomas L. Saaty. „Decision making with the analytic hierarchy process“. In: *International Journal of Services Sciences* 1.1 (2008), S. 83. ISSN: 1753-1446. DOI: 10.1504/IJSSCI.2008.017590.
- [22] Siar Sarferaz, Hrsg. *Compendium on Enterprise Resource Planning: Market, Functional and Conceptual View based on SAP S/4HANA*. 1st ed. 2022. Cham: Springer International Publishing und Imprint: Springer, 2022. ISBN: 978-3-030-93855-0. DOI: 10.1007/978-3-030-93856-7.
- [23] Sensedia, Hrsg. *The Future is Composable: How APIs, Microservices and Events are reshaping the next-gen Enterprises*. 2020. URL: <https://f.hubspotusercontent30.net/hubfs/4209582/%5BInternational%5D%20Boardroom%20Nordics/%28EN%29%20Composable%20Enterprises%20-%20Sensedia.pdf>.
- [24] Software Architekt SAP DTS Integration, Hrsg. *Expertengewichtung 1*. 27.03.2023.
- [25] Joseph T. Vesey. „Time-to-market: Put speed in product development“. In: *Industrial Marketing Management* 21.2 (1992), S. 151–158. ISSN: 0019-8501. DOI: 10.1016/0019-8501(92)90010-Q. URL: <https://www.sciencedirect.com/science/article/pii/001985019290010q>.
- [26] Wolfgang Vieweg. „Agiles (Projekt-)Management“. In: *Management in Komplexität und Unsicherheit*. Springer, Wiesbaden, 2015, S. 41–42. DOI: 10.1007/978-3-658-08250-5_11. URL: https://link.springer.com/chapter/10.1007/978-3-658-08250-5_11.
- [27] Alberto Vivenzio, Hrsg. *Testmanagement Bei SAP-Projekten: Erfolgreich Planen * Steuern * Reporten Bei der Einführung Von SAP-Banking*. 1st ed. Wiesbaden: Springer Fachmedien Wiesbaden GmbH, 2013. ISBN: 978-3-8348-1623-8. DOI: 10.1007/978-3-8348-2142-3.

- [28] Fiorella Zampetti u. a. „CI/CD Pipelines Evolution and Restructuring: A Qualitative and Quantitative Study“. In: *2021 IEEE International Conference 9/27/2021 - 2021*, S. 471–482. DOI: 10.1109/ICSME52107.2021.00048.

Experteninterviews

- [29] Product Owner SAP BTP Prod&Infra und Experte 1, Hrsg. *Experteninterview 1*. 17.03.2023.

Anhang

Anhangsverzeichnis

A	Allgemeine Ergänzungen	XIV
A.1	DevOps	XIV
A.2	Ramped-Deployment	XV
B	Grafiken	XVI
C	Expertenmaterialien	XXIV
C.1	Experteninterview 1	XXV
C.2	Experteninterview 2	XXIX
C.3	Experteninterview 3	XXXII
C.4	Experteninterview 4	XXXIV
C.5	Kodierung der Experteninterviews	XXXVII
C.6	Expertengewichtung 1	XLVIII
C.7	Expertengewichtung 2	LIII
C.8	Expertengewichtung 3	LVIII
C.9	Expertengewichtung 4	LXIII
C.10	Expertengewichtung 5	LXVIII
C.11	Expertengewichtung Durchschnitt	LXXIII

Anhang A Allgemeine Ergänzungen

A.1 DevOps

Prägnant zusammenfassen lässt sich das DevOps-Konzept durch das Akronym CAMS: *Culture (Kultur)*, *Automation (Automatisierung)*, *Measurement (Messung)* und *Sharing (Teilen)* [11, S. 5]. Dabei gilt *Kultur* als das wohl wesentlichste DevOps-Erfolgselement. Diese bezweckt eine Kollaborationsmentalität, welche sich über alle Ebenen eines Unternehmens erstreckt. Operative Entscheidungen sollen dabei auf die Fachebenen herunter delegiert werden, welche aufgrund ihrer spezifischen Expertise am geeignetsten sind, Dispositionen zu verabschieden [11, S. 5]. Eine *Automatisierung* der Softwarebereitstellungsprozesse ermöglicht, sich wiederholende manuelle Arbeit zu eliminieren. Dies kann ebenfalls zur Rationalisierung und damit zur Senkung der IT-Betriebskosten beitragen. Der dabei erzielte Einfluss wird anhand verschiedener DevOps-Kennzahlen bemessen (*Messung*). Neben der Systemverfügbarkeit und der Instandsetzungszeit sind für Softwareentwicklungsunternehmen insbesondere *Time-to-Market* sowie *Time-to-Value* signifikante Metriken [11, S. 7]. Der *Time-to-Market* beschreibt die Zeitspanne zwischen Entwicklungsentstehungsprozess und der Markteinführung von IT-Services [25, S. 141]. Auch der *Time-to-Value* erhält zunehmend Bedeutung in der Softwareentwicklung. Im Gegensatz zum *Time-to-Market* wird hier nicht die Zeit bis zur Komplett-Einführung, sondern das Intervall bis die von dem Softwareunternehmen entwickelte Lösung ersten Kundennutzen herbeiführt, bemessen. Obwohl der im *Time-to-Value* bereitgestellte IT-Service möglicherweise Verbesserungspotenzial besitzt, überwiegt für Kunden des Unternehmens der mit der initialen Auslieferung herbeigeführte Mehrwert. Eine solche Früheinführung ermöglicht dem Softwareunternehmen ebenfalls einen Vorsprung gegenüber Konkurrenten. So ist diesem bereits gelungen, erste Kunden zu akquirieren, deren Input und Feedback möglichst rasch erfasst und verarbeitet werden kann [11, S. 9]. Softwareunternehmen können IT-Services ab dem Pre-Launch somit sukzessive und ressourcenoptimiert unter Zusammenarbeit mit den Pilotkunden erweitern. Auch Adam Caplan, leitender Strategieberater bei Salesforce, emp-

fehlt angesichts der bei Softwareintegration entstehenden Komplexität, Anwendungen schnellst möglichst in produktionsähnlichen Umgebungen zu testen [25]. Aus diesen Erfahrungen sollen Best-Practises entwickelt werden, welche innerhalb von Teams und organisationsübergreifend weitergegeben werden (*Teilen*) [11, S. 7].

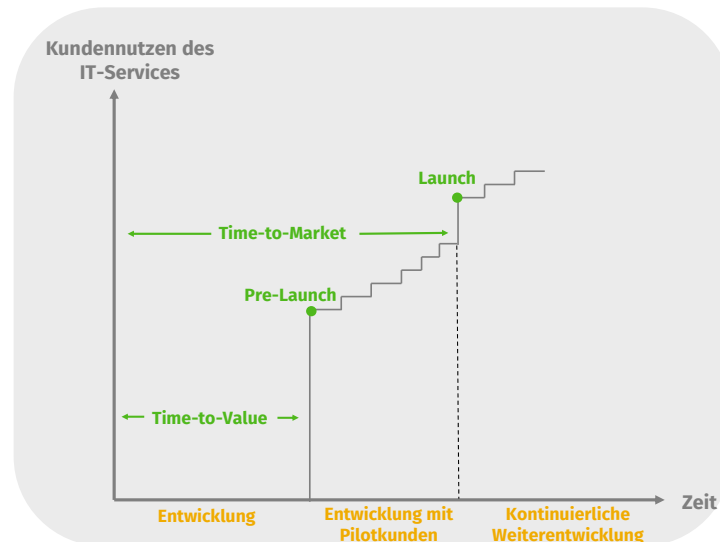


Abbildung 20: Zeitliche Darstellung der Herbeiführung von Kundennutzen bei der Entwicklung von IT-Services [11, S. 9].

A.2 Ramped-Deployment

Für Anwendungen auf einer kritischen IT-Infrastruktur wird i.d.R. die *Ramped-Deployment-Strategie* verwendet. Diese ermöglicht eine präzise Kontrolle horizontal skalierten Services. Bei einer horizontalen Skalierung erfolgt die Replizierung identischer Service-Instanzen, wodurch die Ausfallsicherheit einer Anwendung optimiert werden kann. Die neue Softwareversion wird während des Ramped-Deployment-Prozesses schrittweise auf die horizontalen Instanzen ausgerollt. Dabei werden die ersten aktualisierten Instanzen lediglich für bestimmte Anwender, eine sog. *Ramped-Gruppe*, bereitgestellt. Dabei soll das von dieser Anwendergruppe zur Verfügung gestellte Feedback während zukünftiger Planungsprozesse berücksichtigt werden [0].

Anhang B Grafiken

Stage View

	Prepare	Build	Additional_Tests	Acceptance	Compliance	Deploy
#104 Apr 07 13:17 2 commits	5s	4min 05s	2min 16s	7min 48s	1min 51s	2min 59s
#103 Apr 07 12:40 2 commits	6s	4min 13s	2min 08s			

Abbildung 21: Integration- und Delivery-Zeit für den RiskService mit Jenkins. Eigene Darstellung.

Stage View

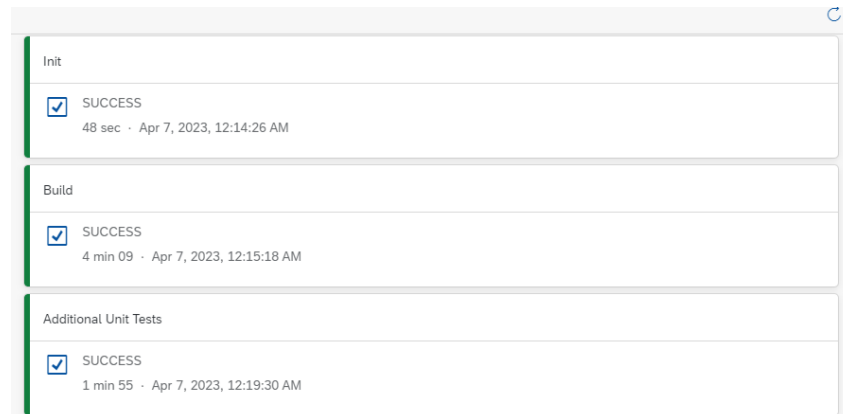
	Prepare	Build	Additional_Tests	Acceptance	Compliance	Deploy
#99 Apr 07 11:46 2 commits	5s	3min 58s	2min 5s	7min 35s	1min 46s	2min 53s
#98 Apr 07 11:10 6 commits	6s	4min 3s	1min 55s			

Abbildung 22: Integration- und Delivery-Zeit für den SupplierService mit Jenkins. Eigene Darstellung.

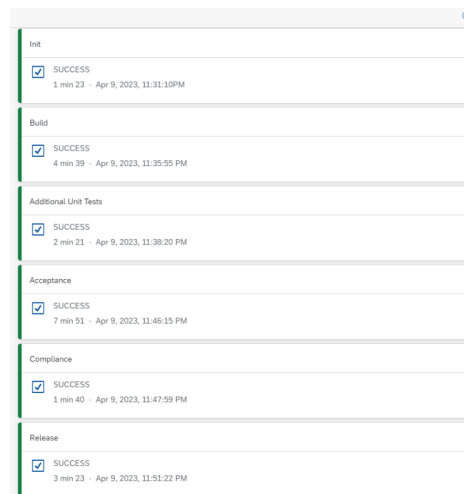
Stage View

	Prepare	Build	Additional_Tests	Acceptance	Compliance	Deploy
#111 Apr 07 14:52 2 commits	6s	4min 06s	2min 14s	7min 53s	1min 53s	2min 58s
#110 Apr 07 14:16 2 commits	6s	4min 7s	2min 10s			

Abbildung 23: Integration- und Delivery-Zeit für den PlantService mit Jenkins. Eigene Darstellung.



Init
<input checked="" type="checkbox"/> SUCCESS 48 sec · Apr 7, 2023, 12:14:26 AM
Build
<input checked="" type="checkbox"/> SUCCESS 4 min 09 · Apr 7, 2023, 12:15:18 AM
Additional Unit Tests
<input checked="" type="checkbox"/> SUCCESS 1 min 55 · Apr 7, 2023, 12:19:30 AM



Init
<input checked="" type="checkbox"/> SUCCESS 1 min 23 · Apr 9, 2023, 11:31:10PM
Build
<input checked="" type="checkbox"/> SUCCESS 4 min 39 · Apr 9, 2023, 11:35:55 PM
Additional Unit Tests
<input checked="" type="checkbox"/> SUCCESS 2 min 21 · Apr 9, 2023, 11:38:20 PM
Acceptance
<input checked="" type="checkbox"/> SUCCESS 7 min 51 · Apr 9, 2023, 11:46:15 PM
Compliance
<input checked="" type="checkbox"/> SUCCESS 1 min 40 · Apr 9, 2023, 11:47:59 PM
Release
<input checked="" type="checkbox"/> SUCCESS 3 min 23 · Apr 9, 2023, 11:51:22 PM

Abbildung 24: Integration- und Delivery-Zeit für den RiskService mit SAP CI/CD. Eigene Darstellung.

Init	
<input checked="" type="checkbox"/>	SUCCESS 48 sec · Apr 7, 2023, 11:38:26 AM
Build	
<input checked="" type="checkbox"/>	SUCCESS 4 min 07 · Apr 7, 2023, 11:39:15 AM
Additional Unit Tests	
<input checked="" type="checkbox"/>	SUCCESS 1 min 59 · Apr 7, 2023, 11:43:23 AM

Init	
<input checked="" type="checkbox"/>	SUCCESS 1 min 19 · Apr 9, 2023, 11:04:00 PM
Build	
<input checked="" type="checkbox"/>	SUCCESS 4 min 32 · Apr 9, 2023, 11:05:20 PM
Additional Unit Tests	
<input checked="" type="checkbox"/>	SUCCESS 2 min 15 · Apr 9, 2023, 11:09:52 PM
Acceptance	
<input checked="" type="checkbox"/>	SUCCESS 7 min 55 · Apr 9, 2023, 11:12:08 PM
Compliance	
<input checked="" type="checkbox"/>	SUCCESS 1 min 29 · Apr 9, 2023, 11:20:04 PM
Release	
<input checked="" type="checkbox"/>	SUCCESS 3 min 18 · Apr 9, 2023, 11:21:33 PM
Declarative: Post Actions	
<input checked="" type="checkbox"/>	SUCCESS 90 ms · Apr 9, 2023, 11:24:52 PM

Abbildung 25: Integration- und Delivery-Zeit für den SupplierService mit SAP CI/CD. Eigene Darstellung.

Init
<input checked="" type="checkbox"/> SUCCESS 47 sec · Apr 7, 2023, 12:32:02 AM
Build
<input checked="" type="checkbox"/> SUCCESS 4 min 11 · Apr 7, 2023, 12:32:51 AM
Additional Unit Tests
<input checked="" type="checkbox"/> SUCCESS 1 min 59 · Apr 7, 2023, 12:37:15 AM

Init
<input checked="" type="checkbox"/> SUCCESS 1 min 21 · Apr 10, 2023, 06:04:12 PM
Build
<input checked="" type="checkbox"/> SUCCESS 4 min 42 · Apr 10, 2023, 06:05:35 PM
Additional Unit Tests
<input checked="" type="checkbox"/> SUCCESS 2 min 23 · Apr 10, 2023, 06:10:19 PM
Acceptance
<input checked="" type="checkbox"/> SUCCESS 7 min 48 · Apr 10, 2023, 06:12:45 PM
Compliance
<input checked="" type="checkbox"/> SUCCESS 1 min 43 · Apr 10, 2023, 06:20:40 PM
Release
<input checked="" type="checkbox"/> SUCCESS 3 min 29 · Apr 10, 2023, 06:22:25 PM

Abbildung 26: Integration- und Delivery-Zeit für den PlantService mit SAP CI/CD. Eigene Darstellung.

Anhang C Expertenmaterialien

C.1 Experteninterview 1

Interviewpartner: Product Owner SAP BTP Prod&Infra (Experte 1)

Datum: 17.03.2023

Interview-Medium: Microsoft-Teams

1 **Interviewer:** Du kannst dich ja mal kurz vorstellen und erläutern, was du be-
2 reits mit dem CI/CD-Bereich zu tun hattest und was deine täglichen Aufgaben sind.

3 **Experte:** Ich bin Product Owner für den Continuous Integration and Delivery Ser-
4 vice. Meine tägliche Aufgabe ist die Steuerung des Backlogs für unsere Anforderun-
5 gen. Dabei muss ich die Anforderungen, die über verschiedene Kanäle von unseren
6 Kunden hereinkommen konsolidieren und für unsere Abteilung bereitstellen.

7 **Interviewer:** Wie definierst du den Begriff CI/CD?

8 **Experte:** Also für mich gibt es einmal den CI Begriff. Dabei habe ich einen CI-
9 Server, der mir nach einem Push in mein zentrales Repository innerhalb kurzer Zeit
10 ein Feedback gibt. Danach kommt der CD-Prozess. Dabei kann ich abhängig von
11 verschiedenen Mechanismen, wie zum Beispiel ein Review oder einem Request die
12 CD-Pipeline auslösen. Die ist dann auch mächtiger als die CI-Pipeline. Mit dieser
13 wird zentral gebaut, getestet und gegebenenfalls auch noch Sachen wie Complian-
14 ce, Vulnerabilities, statische Codechecks, Integrations-Tests und Performance-Tests
15 abgewickelt. Das getestete Programm kann dann anschließend zum Beispiel in ein
16 Artefakt-Repository oder in eine Produktionsumgebung bereitgestellt werden.

17 **Interviewer:** Welche Vorteile hat es, wenn Software kontinuierlich bereitgestellt
18 wird?

19 **Experte:** Der Vorteil ist der, dass ich meine Änderungen in kleinen Paketen, die
20 sich auch leichter integrieren lassen, mache. Wenn ich tägliche oder alle zwei drei
21 Tage Changes mache und dann jeweils schaue, ob der Status noch grün ist, birgt
22 das gegenüber dem klassischen Wasserfallmodell sehr viele Vorteile. Wenn ich dann
23 schnell in eine Canary-Umgebung deploye, kann ich natürlich früh Fehler finden,
24 was schließlich auch deutlich günstiger ist.

25 **Interviewer:** Welche unterschiedlichen Arten von Pipelines gibt es?

26 **Experte:** Das hängt ein wenig von den Anforderungen. Also typischerweise hat
27 man eine sehr kleine Pipeline für Request-Votes. Die wird automatisch ausgelöst,
28 wenn in dem Github ein Pull-Request aufgemacht wird. Die sollte dann maximal 10
29 - 15 Minuten laufen. So soll der Entwickler ein schnelles Feedback bekommen. Dann
30 gibt es da noch die Delivery-Pipelines. Dazu gehört, wenn man ein Artefakt in ein
31 Repository deployed oder man auch tatsächlich releaset. Für solche Pipelines kann
32 entschieden werden, ob entweder alles am Stück gemacht wird oder ob Komponenten
33 aufgeteilt werden. Also, dass ich am Anfang ein paar kleine Unit- und Code-Tests
34 mache und das in mein Artefakt Repository bereitstelle. Zu einem gegebenen Zeit-
35 punkt x kann ich dann sagen, dass ich entsprechend aufwändigere Tests mache.

36 **Interviewer:** Du hattest Artefakt-Repository genannt. Welche Vorteile bringt das?

37 **Experte:** Das spielt gerade in der Composable Enterprise Architektur eine wichtige
38 Rolle. Kleine entwickelte Komponenten können mit Versionierung in das Artefakt-
39 Repository bereitgestellt werden. Andere Entwickler können diese Komponente dann
40 aus dem Artefakt-Repository herausziehen und für eigenen Entwicklungen wieder-
41 verwenden.

42 **Interviewer:** Welche Stages hat eine typische CI/CD-Pipeline?

43 **Experte:** Typischerweise beginnt es mit der Build-Stage, bei welcher Unit-Tests
44 ausgeführt werden. Für CAP werden dabei die Frameworks Mocha oder Jest ver-
45 wendet. Dann haben wir eine Acceptance-Stage, in welcher dann Akzeptanztests
46 laufen. Solche Akzeptanztests können dann z.B. auch Integration-Tests umfassen.
47 Die Integration-Tests werden mit Newman gemacht. Dann gibt es eine Compliance-
48 Stage. In dieser laufen dann Tools wie die SonarQube. Dort wird dann z.B. geprüft,
49 ob ich irgendwelche Lizenzrechte verletze. Dann kommt die Security-Stage. Dort
50 wird nach Vulnerabilities und statischen Kontexten geprüft und evaluiert ob Ge-
51 fahr für Cross-Skripting Null-Pointer-Exceptions etc. besteht. Dann gibt es noch die
52 Release-Stage. Dort wird die Anwendung dann tatsächlich in die Cloud-Plattform
53 bereitgestellt. Es benötigt einer Unterstützung dieser Stufen, um eine effiziente Be-
54 reitstellung von Software zu ermöglichen.

55 **Interviewer:** Welche Pipelines werden denn so bei der SAP verwendet?

56 **Experte:** Zum einen wird der von der SAP bereitgestellte CI/CD-Service verwen-
57 det. Des Weiteren gibt es Jenkins. Diese wird i.d.R. mit Project Piper verwendet.
58 Die Jenkins muss dabei selbst gehostet werden. Für interne Projekte wird dafür das
59 Jenkins-as-a-Service angeboten. Häufig wird für interne Projekte auch Azure De-
60 vOps verwendet.

61 **Interviewer:** Welche Aspekte sind bei der Wahl eines CI/CD-Pipeline-Tools zu be-
62 achten?

63 **Experte:** Zum einen wie viel Wissen habe ich denn jetzt schon. Da sollte evaluiert
64 werden, ob man DevOps-Spezialisten hat die schon häufig Pipelines implementiert
65 haben. Für Abteilungen welche keine DevOps-Spezialisten haben spielt die Benutzer-
66 freundlichkeit eine große Rolle. Da ist es zum einen wichtig, wie leicht sich die Tools
67 warten lassen, aber auch wie leicht sich eine Pipeline implementieren lässt. Weiterhin
68 ist wichtig zu wissen, wie flexibel man bei der Pipeline-Gestaltung sein will. Zudem
69 muss natürlich auch evaluiert werden, welche Funktionalitäten also Tests, Code-
70 Scans und Builds auf der Pipeline ausgeführt werden sollen. Zuletzt sollte dann was
71 die Funktionalität angeht auch noch evaluiert werden auf welcher Plattform die Soft-
72 ware bereitgestellt werden soll. Skalierbarkeit spielt dann auch noch eine wichtige
73 Rolle. Horizontale Skalierbarkeit umschreibt in diesem Kontext, dass mehrere Build
74 gleichzeitig durchgeführt werden können. Die vertikale Skalierbarkeit bedeutet, dass
75 die Ressourcen einer Pipeline-Instanz flexibel angepasst werden können. Da Jenkins
76 selbst gehostet wird, hat das natürlich in diesem Aspekt einen großen Nachteil. Für
77 unsere Kunden spielt auch die Sicherheit eine wichtige Rolle. Diese ist essenziell,
78 damit in die Produktionsumgebungen keine Maleware eingeschläust wird.

79 **Interviewer:** Wie sieht es mit der Unterstützung von Tests aus?

80 **Experte:** Es ist eigentlich fast alles auf dem SAP BTP CI/CD-Service möglich.
81 Was bisher noch nicht wirklich funktioniert sind API-Tests.

82 **Interviewer:** Wie sieht es mit den Integrationsmöglichkeiten aus. Worauf muss da
83 geachtet werden?

84 **Experte:** Integration ist auch ein sehr wichtiger Aspekt bei der Auswahl von CI/CD-

85 Pipelines. Da muss natürlich auf die Integrationsmöglichkeiten mit dem Repository
86 geachtet werden. Der SAP CI/CD-Service unterstützt dabei einen ganz normalen
87 Git-Server. Was auch noch funktioniert sind BitBucket Repositories. Die Integration
88 funktioniert dabei über eine Webhook. Es können dabei jedoch ausschließlich Com-
89 mit Events verarbeitet. Sehr selten wird eine CI/CD-Pipeline auch in die Entwick-
90 lungsumgebung integriert. Aber das ist mit dem SAP CI/CD-Service nicht möglich.
91 Was auch gemacht wird, aber eher seltener ist, die Pipeline in die Entwicklungsum-
92 gebung zu integrieren. SAP CI/CD hat diese Möglichkeit nicht. Jenkins und Azure
93 können dabei sowohl in Eclipse als auch VSCode eingebunden werden.

94 **Interviewer:** Gibt es irgendwelche Einschränkungen bezüglich der Laufzeitumge-
95 bung?

96 **Experte:** Bei unserem Service nicht. Wir können sowohl auf Cloud Foundry als
97 auch auf Kyma deployen.

98 **Interviewer:** Gibt es in dem SAP CI/CD-Tool irgendwelche Überwachungsfunktionalitäten?

99 **Experte:** Für die Anwendung selbst gibt es direkt Funktionalitäten auf der BTP.
100 Da gibt es verschiedene Notification-Service, die über den Erfolg der Pipeline-Builds
101 benachrichtigen. Aber ein direktes Monitoring der Pipeline gibt es nicht.

102 **Interviewer:** Welche Kosten fallen für die CI/CD-Pipeline an?

103 **Experte:** Eine Build-Hour kostet einen Euro.

104 **Interviewer:** Sind parallele Builds möglich und ist die Pipeline mit dem Transport
105 Management System integrierbar?

106 **Experte:** Nein leider nicht. Aber die Pipeline kann Software auf das Transport Ma-
107 nagement System bereitstellen.

108 **Interviewer:** Welche Support-Möglichkeiten stehen für SAP CI/CD bereit?

109 **Experte:** Wir bieten wie andere SAP-Cloud-Dienste ein Ticket-System mit Service-
110 Now an. Wenn Kunden konkrete technische Unterstützung benötigen, können sich
111 diese an die technischen Berater der SAP wenden.

112

C.2 Experteninterview 2

Interviewpartner: Product Manager SAP Hyperspace CI/CD (Experte 2)

1 Datum: 24.03.2023

2 Interview-Medium: Microsoft-Teams

3 **Interviewer:** Du kannst dich nun gerne vorstellen. Wie kommst du während deiner
4 Arbeit mit CI/CD in Verbindung?

5 **Experte:** Ich bin Product Manager. Ich habe zuerst für den SAP BTP CI/CD-
6 Service gearbeitet. Nun mit ich im Hyperspace.

7 **Interviewer:** Was bedeutet für dich CI/CD?

8 **Experte:** CI ist die Integration bei welchem die Änderungen von unterschiedlichen
9 Entwickler so schnell wie möglich in einem Source-Code-Management-System in-
10 tegriert werden müssen. CD ist Continuous Delivery. Das ist die Möglichkeit ein
11 Feature so schnell wie möglich auf die Produktion zu deployen und für den Kunden
12 bereitzustellen

13 **Interviewer:** Welchen Vorteil hat es Software schnell bereitzustellen?

14 **Experte:** Der Vorteil für mich ist, dass man mit kleineren Pakete arbeitet. So
15 ist die Gefahr, dass etwas im Produktivsystem kaputtgeht sehr gering. Mit klei-
16 nen Änderungen sind die Auswirkungen, die eine Integration hat auch besser zu
17 überblicken.

18 **Interviewer:** Aus welchen typischen Komponenten besteht eine gewöhnliche Pipe-
19 line?

20 **Experte:** Also man fängt typischerweise mit dem Sync auf seinem Git Repository
21 an. Der zweite Schritt ist dann der Build. Dort werden dann auch Unit-, Integration-,
22 und Acceptance-Tests in unterschiedlichen Spaces ausgeführt. In einer Acceptance-
23 Stage, fließen dann noch mehr Teile zusammen. Dazu gehören dann neben normalen
24 Tests auch Security-Scans. Zuletzt wird die Software dann deployet. **Interviewer:**
25 Wie sind Pipelines aufgebaut?

26 **Experte:** Also früher haben wir keine unterschiedlichen Pipelines für CI und CD
27 verwendet. Da haben wir aber gesehen, dass das ziemlich problematisch ist. Wenn

28 alles sequenziell ausgeführt wird und dann in der Mitte irgendwas abbricht, dann
29 haben wir sehr viel Zeit verloren. Nun geht der Trend in Richtung Shift-Left. Die
30 Pipelines werden somit deutlich verkleinert und somit bekommt man auch schneller
31 Feedback.

32 **Interviewer:** Welche Kriterien sollte man bei der Auswahl von Pipelines beachten.

33 **Experte:** Es kommt natürlich darauf an, welche Produktstandards vorgegeben sind.
34 Wenn die Standards hoch sind, dann ist die Test-Funktionalität sehr wichtig. Da-
35 zu gehören dann, insbesondere Security-Checks, wie mit Fortify. Wenn du in sehr
36 großen Entwicklungen bist, dann ist es natürlich auch sehr wichtig, dass die Pipeline
37 eine gute Performance besitzt. Somit kann Software schneller bereitgestellt werden.
38 Des Weiteren ist es essenziell, dass die CI/CD-Prozesse überwacht werden können.
39 Gerade bei komplexen Systemen mit vielen Services ist das für DevOps-Engineers
40 oft die einzige Möglichkeit die Prozesse nachhaltig zu überblicken. Dann ist natürlich
41 auch wichtig, wie gut sich die Pipeline in die Infrastruktur integrieren. Bei dieser
42 Integration ist dann auch wichtig, ob alle Sicherheitsstandards eingehalten werden.
43 Insbesondere für kleinere Kunden ist es natürlich auch essenziell, wie viel die Pi-
44 peline kostet. Für viele Kunden ist darüber hinaus der Support wichtig. Es sollten
45 dabei kontinuierliche Updates, Schulungsmaterial sowie wie eine gute Dokumentation
46 verfügbar sein. Außerdem ist es für Entwickler immer vorteilhaft, wenn für die Tools
47 eine große Community existiert.

48 **Interviewer:** Mit welchen Pipelines hattest du bisher Erfahrung?

49 **Experte:** Mit Azure DevOps habe ich bisher noch keine Erfahrung gemacht. In
50 meinem jetzigen Team arbeite ich mit dem Jenkins-as-a-Service. In meinem vorhe-
51 rigen Team habe ich auch Erfahrung mit dem SAP BTP CI/CD-Service gemacht.
52 Ursprünglich wurde das Projekt Piper für Jenkins nur für interne Projekte genutzt.
53 Mittlerweile wurde die Bibliothek als Open-Source veröffentlicht. Für interne Projek-
54 te darf der SAP BTP CI/CD aufgrund der derzeitigen Produktstandards nicht ver-
55 wendet werden. Dieser wird eigentlich nur für Kunden angeboten. Das SAP CI/CD-
56 Tool lohnt sich insbesondere für Kunden, die noch nicht viel DevOps-Expertise be-
57 sitzen und auch keine teure Infrastruktur betreiben wollen.

58 **Interviewer:** Ist in dem CI/CD-Service von der SAP schon der Preis für Tools wie
59 SonarQube mit einberechnet?

60 **Experte:** Nein der Preis ist nicht mit drin. Tools wie SonarQube müssen von den
61 Kunden selbst gehostet werden.

62 **Interviewer:** Weißt du ob die Tools in das SAP CTM integrierbar sind?

63 **Experte:** Ja sowohl JaaS als auch SAP BTP CI/CD sind in das SAP CTM inte-
64 grierbar. Intern habe ich noch nicht oft gehört, dass dieser verwendet wird. Aber
65 Kunden können theoretisch auf die CTM bereitstellen. Das CTM ist ebenfalls mit
66 einem Change Management Surface verbunden. Damit kann man einen Change-
67 Auftrag erstellen und dann Artefakte zwischen unterschiedlichen Systemen bereit-
68 stellen. Dadurch hat man einfach mehr Transparenz. Außerdem bietet das System
69 für Composable-ERP-Systeme einen großen Vorteil. Über das SAP CTM können
70 Abhängigkeiten zwischen verschiedenen Microservices definiert werden.

71 **Interviewer:** Welche Überwachungsfunktionalitäten bieten die Pipelines und wel-
72 che Tools werden von Kunden in der Regel verwendet?

73 **Experte:** Bei Jenkins weiß ich, dass man Logs auslesen kann und den Workflow
74 somit nachvollziehen kann. Innerhalb der SAP wird i.d.R. das SAP-Partner-Tool
75 Splunk verwendet. Das ist über Project Piper einbindbar. Kunden nutzen häufig
76 auch andere Open-Source-Tools. Ein sehr häufig verwendetes Tool ist dabei das
77 Kibana-Dashboard.

C.3 Experteninterview 3

Interviewpartner: Product Manager SAP Hyperspace Security Tools (Experte 3)

Datum: 22.03.2023

Interview-Medium: Microsoft-Teams

1 **Interviewer:** Wie hat sich das Thema Security im CI/CD-Kontext verändert?

2 **Experte:** In letzter Zeit hat sich das Thema Shift Left sehr stark etabliert. Das be-
3 deutet, dass das Feedback eigentlich möglichst früh an den Entwickler zurückgegeben
4 wird. Der Entwickler lernt somit viel nachhaltiger, da er im Integration-Kontext noch
5 keine große Verantwortung hat und somit kleine Arbeitspakete zur Verfügung gestellt
6 bekommt. Wenn du hingegen ganz große Pakete in die Main-Line integrierst dann
7 ist irgendwann nicht mehr ersichtlich wer welche Änderungen gemacht hat. Früher
8 gab es dabei immer ein Security-Experten der sich vor der Auslieferung dann um
9 alles kümmern musste.

10 **Interviewer:** Wie wird Security in DevOps heutzutage gemacht?

11 **Experte:** Security sollte nicht mehr nur von einem Spezialisten behandelt werden.
12 Vielmehr sollte dies als Kollektiv abgehandelt werden. Jeder muss bei der Entwick-
13 lung seiner Funktionalitäten schon so früh wie möglich schauen, ob alle sicherheitsre-
14levanten Aspekte eingehalten wurden. Das wird dann i.d.R. durch Automatisierung
15 gemacht. Es wird dabei schon sehr lange mit Security-Tools gearbeitet. Diese sind
16 aber nicht sehr benutzerfreundlich. D.h., dass die Findings nicht gut präsentiert wer-
17 den. Somit versteht ein normaler Entwickler nicht, was mit einem Finding gemeint
18 ist und wie dieses Problem behoben werden kann. Da haben sich in der letzten Zeit
19 aber sehr viele neue und benutzerfreundlichere Tools etabliert. Diese sollen dabei
20 helfen den Shift-Left-Ansatz voranzutreiben.

21 **Interviewer:** Welche Arten von Security-Tools gibt es

22 **Experte:** Es gibt i.d.R. zwei verschiedene Arten von Security-Tools. Es gibt da zum
23 einen die statischen Code-Analysen (SAST). Dort wird insbesondere OS-Scanning
24 betrieben. Dann gibt es noch das Dynamic Application Security Testing. Dort wer-

25 den dann auch tatsächlich UI-Elemente, APIs sowie Datenbanken gescannt. Somit
26 können dann z.B. Scripting-Attacks oder SQL-Injections verhindert werden. Manch-
27 mal wird dann auch noch die Kategorie des Interactive Application Security Testing
28 (IAST) definiert. Dabei wird ein Agent in die Laufzeitumgebung mit integriert, wel-
29 cher die Insights der Analysen liefert. Dort können dann z.B. Software-Component-
30 Analysen gemacht werden. Dabei werden HTTP-Requests gespooft, um bestimmte
31 Lücken im System zu finden.

32 **Interviewer:** Welche Tools werden bei der SAP mit Project Piper angewendet?

33 **Experte:** Da haben wir z.B. Fortify. Das ist insbesondere für Java und Python. Das
34 Tool ist allerdings kaum noch in Verwendung, da es sich historisch nicht weiterent-
35 wickelt hat. In näherer Zukunft wird das durch GitHub Advanced Security abgelöst.
36 Für CAP Node wird von in den Produktstandards das Tool Checkmarx vorgeschrie-
37 ben. Für Open-Source ist das Tool Whitesource vorgeschrieben. Für SAP UI5 wird
38 bei der statischen Code-Analyse Checkmarx verwendet. Für Open-Source gibt es kei-
39 ne Vorgabe. Das liegt daran, dass UI5 eigentlich über JavaScript geschrieben wird
40 und somit NPM als Package-Manager bräuchte. Node wird in dieser Technologie
41 jedoch nicht unterstützt. Für statische Codeanalysen wird SonarQube verwendet.

C.4 Experteninterview 4

Interviewpartner: Frontend Test Developer SAP Hyperspace Adoption & Onboarding (Exper

Datum: 22.03.2023

Interview-Medium: Microsoft-Teams

1 **Interviewer:** Du kannst dich nun gerne vorstellen.

2 **Experte:** Derzeit bin ich im Adoption and Onboarding Team von Hyperspace. Wir
3 unterstützen Kunden darin ihre Projekte zu onboarden. Dafür bieten wird verschie-
4 dene Toolings, wie Security-Tools, Deployment-Tools, Test-Tools etc. an.

5 **Interviewer:** Was hat das Hyperspace denn mit CI/CD zu tun?

6 **Experte:** Hyperspace ist eine Plattform, die einem ermöglicht, sein CI/CD-Setup
7 möglichst konsistent und einfach aufzusetzen. Das soll einem die Möglichkeit geben
8 den kognitiven Load in den Teams zu reduzieren. So muss nicht alles manuell ge-
9 macht werden. So ein Aufsetzen von einer Pipeline mit Jenkins und Groovy-Scripten
10 usw. kann ja ein ziemlicher Aufwand sein. Da benötigt man sehr viel Wissen. Hy-
11 perspace gibt den Entwicklungsteams Guidelines vor. D.h. auf Hyperspace kann ich
12 einfach ein Template auswählen. Das Hyperspace kümmert sich dann darum, das
13 alle benötigten Tools zur Verfügung stellen, dass du da nicht explizit in jedem Tool
14 alles wieder selbst konfigurieren musst.

15 **Interviewer:** Wird im Hyperspace auch eine konkrete Step-Implementierung abge-
16 nommen?

17 **Experte:** Hyperspace erzeugt einem eigentlich erst mal so eine Ready-Made-Pipeline.
18 Das ist eine standardisierte Vorgabe auf welcher man dann Konfiguration vornimmt.
19 Dann kannst du z.B. einstellen welche Tools du verwenden möchtest. Du kannst
20 natürlich davon ausbrechen und sagen okay, ich möchte da jetzt einen kompletten
21 Step überschreiben. Das Ziel ist jedoch den kognitiven Load so gering wie möglich
22 zu halten.

23 **Interviewer:** Wie definierst du für dich CI/CD?

24 **Experte:** Ich bin dort jetzt kein kompletter Experte, aber mein Hauptmotiv als

25 Entwickler ist es möglichst schnelles Feedback zubekommen. Früher war es so, dass
26 ich Tests geschrieben habe und die dann einmal in der Woche ausgeführt habe. Auf-
27 grund der Verzögerung ist das natürlich nicht besonders geschickt. Bei einem guten
28 Setup mache ich eine Änderung und bekomme beim Commit direkt ein Feedback.
29 Das zweite ist natürlich, dass ich neue Produktversionen sehr schnell zum Kunden
30 bekomme. Idealerweise innerhalb von einer Woche oder vielleicht sogar manchmal
31 in einem Tag. Als ich damals in einer SAP-Partnerfirma war, haben wir manchmal
32 ein ganzes Jahr entwickelt. Dann gab es eine sehr große Testphase. Und am Ende
33 hat sich herausgestellt, dass der Kunde etwas ganz anderes haben wollte.

34 **Interviewer:** Welchen Vorteil hat es, wenn man den CI/CD-Prozess automatisiert?

35 **Experte:** Man hat die Möglichkeit neue Features erstmal einzelnen Kunden bereit-
36 zustellen. Wenn ich dann feststelle, dass irgendwas nicht funktioniert kann ich schnell
37 ein Rollback machen. Da gibt es dann z.B. das Feature-Toggle. Da wird eine neue
38 Funktionalität dann hinter einer Flag versteckt. Wenn ein bestimmter Kunde dieses
39 Feature dann haben möchte, dann setzt er entsprechend die Flag.

40 **Interviewer:** Welche Art von Pipelines werden denn i.d.R. verwendet?

41 **Experte:** Ich kenne hauptsächlich die Pull-Request-Pipeline. Häufig gibt es dann
42 auch noch einmal eine Pipeline, welche einmal am Tag läuft, bei welcher dann Tests
43 gemacht werden, welche deutlich länger laufen.

44 **Interviewer:** Welchen Vorteil hat ein Artefakt-Repository?

45 **Experte:** Das Artefakt-Repository wird verwendet, um das Coding was erzeugt
46 wurde versioniert abzulegen. Dieses wird dann in der Pipeline immer wieder verwen-
47 det, um z.B. Tests dagegen auszuführen. Mit diesen Artefakts kann man dann auch
48 noch sehr gut Rollbacks ausführen. Das heißt, wenn man merkt, dass eine neue Ver-
49 sion nicht funktioniert, kann man einfach wieder zur alten Version zurückspringen.

50 **Interviewer:** Welche Pipelines werden bei der SAP im Regelfall verwendet?

51 **Experte:** Auf der Orchestratorseite ist es so, dass ganz viel über Azure-DevOps
52 gemacht wird. Hierbei gibt es jetzt auch einige speziellen Governace-Checks, die
53 darüber möglich sind. Außerdem ist der Wartungsaufwand einfach viel geringer.
54 Das SAP Tools Team hatte dann alles auf einen zentralen Blick und konnte ent-

55 sprechend sagen, dass wenn für eine Pipeline mehr Kapazität benötigt wird, dass
56 entsprechend Ressourcen zugeschaltet werden. Und die SAP hat da wahrscheinlich
57 einfach auch gute Konditionen bekommen. Andere kleine Teams, wie z.B. das Sports
58 One haben dabei eher eine eigene Pipeline über den JaaS. Aber es ist einfach sehr
59 davon abhängig, welche Technologie du hast.

60 **Interviewer:** Welche Tests werden bei der SAP gemacht?

61 **Experte:** Bei der SAP werden durch den Produktstandard gemäß ISO 9001 ver-
62 schiedene Validierungen vorgeschrieben. Für Unit Tests gibt es dafür verschiedene
63 Frameworks. I.d.R. wird bei der SAP Q-Unit für Frontend und Mocha oder Jest für
64 das Backend verwendet. Für Q-Unit wird die Laufzeitumgebung Karma
65 und für Mocha und Jest Node benötigt. Für Frontend-Integration-Tests wird OPA5
66 verwendet. Da wird dann sowas wie die Backend-Anbindung einfach gemockt. Dafür
67 benötigt es ebenfalls der Laufzeitumgebung Karma. Für Integration-Tests im Ba-
68 ckend wird Newman verwendet, mit welcher Postmann-Tests automatisiert werden
69 können. Und für E2E-Tests im Frontend wird dann i.d.R. WDI5 verwendet. Das
70 kann man dafür verwenden, wenn man jetzt tatsächlich eine gesamte App testet.
71 Das hat den Vorteil, dass ich wie ein End-User teste. Nachteil ist dabei jedoch, dass
72 ich schauen muss, dass die Daten verfügbar sind, Customizings gemacht wurden
73 etc. Um OPA5 in einer Pipeline zu automatisieren wird die Webdriver.io Laufzeit-
74 umgebung benötigt. Alle die hier genannten Laufzeitumgebungen werden durch die
75 Project Piper ausgeliefert. Aber man muss die Tests natürlich immer gezielt einsetz-
76 ten. Wir haben damals in unsere Pipeline E2E-Tests eingebaut und dadurch hat die
77 Pipeline um den Faktor 3 länger gebraucht. Noch einmal zur zentralen Aussage der
78 Testpyramide. Die Empfehlung ist möglichst viel auf den unteren Ebenen abzude-
79 cken, also mit Unit-Tests und auf den oberen Ebenen nur noch das zu testen, was
80 man nicht mit Unit- und Integration-Tests testen kann.

81 **Interviewer:** Werden denn immer alle Tests ausgeführt?

82 **Experte:** Also bei einer Pull-Request-Pipeline sollten auf jeden Fall die Unit- und
83 Integration-Tests laufen. Die System-Tests werden dann z.B. einmal am Tag aus-
84 geführt. Manche Teams verwenden auch eine parallele Ausführung von Tests und

85 führen dann eben verschiedene Szenarien gleichzeitig durch. Das läuft dann i.d.R.
86 schneller und dann können solche Tests auch beim Pull-Request ausgeführt werden.
87 Gerade die Compliance und Accessibility-Tests werden dann eher in der Delivery-
88 Pipeline durchgeführt.

89 **Interviewer:** Auf welche Integrationsaspekte muss geachtet werden?

90 **Experte:** In den bisherigen Projekten, in welchen ich gearbeitet habe, wurde die
91 Auswahl einer CI/CD-Pipeline immer sehr stark von dem Repository abhängig ge-
92 macht. Die Repositories, welche von Kunden dabei besonders oft verwendet werden
93 sind GitHub, GitLab und BitBucket. Was auch aber eher selten in Kundenprojekten
94 beachtet wird, ist die Integrierbarkeit in Projektmanagement-Tools. Häufig wird da-
95 bei Jira verwendet, aber i.d.R. machen die Kunden die Wahl einer CI/CD-Pipeline
96 nicht von der Unterstützung eines spezifischen Tools abhängig. Um einen Überblick
97 über die Bereitstellungsprozesse zu erhalten, müssen Experten somit nicht in den
98 Code schauen, sondern haben alles in einem zentralen Tool. SAP CI/CD unterstützt
99 das nicht wohingegen ich gehört habe, dass es diese Möglichkeit bei Jenkins sowie
100 Azure Pipelines gibt.

101 **Interviewer:** Wie wird der Entwickler über den Erfolg der Tests informiert?

102 **Experte:** Da gibt es unterschiedliche Verfahrensweisen. Es gibt da dann z.B. ver-
103 schiedene Monitoring-Tools in der CI/CD-Pipeline. Ein anderer Weg ist, wenn man
104 die CI/CD-Pipeline über APIs in das Repository integriert. Was auch häufig ge-
105 macht wird ist, dass man die Pipelines in den SAP Alert Service integriert, sodass
106 Entwickler dann entsprechend Nachrichten über Mail oder Slack bekommen.

C.5 Kodierung der Experteninterviews

Was ist CI/CD?

Aussage	Kodierung	Experte	Zeilennummer
---------	-----------	---------	--------------

„Dabei habe ich einen CI-Server, der mir nach einem Push in mein zentrales Repository innerhalb kurzer Zeit ein Feedback gibt.“	CI	Experte 1	8 ff.
„Das ist die Möglichkeit ein Feature so schnell wie möglich auf Produktion zu deployieren und für den Kunden bereitzustellen“	CD	Experte 2	10 ff.

Verschiedene Arten von Pipelines

Aussage	Kodierung	Experte	Zeilennummer
„[Mit der CD-Pipeline] wird zentral gebaut, getestet und gegebenenfalls auch noch Sachen wie Compliance, Vulnerabilities, statische Codechecks, Integrations-Tests und Performance-Tests abgewickelt.“	Bestandteile CD-Pipeline	Experte 1	12 ff.
„Die sollte dann maximal 10 - 15 Minuten laufen. So soll der Entwickler ein schnelles Feedback bekommen.“	Pull-Request-Pipeline	Experte 1	26 ff.
„ Die Pipelines werden somit deutlich verkleinert, aber damit bekommt man auch schneller Feedback.“	Kleine Pipelines	Experte 2	29 ff.

Deploy und Release

Aussage	Kodierung	Experte	Zeilennummer
„Das getestete Programm kann dann anschließend zum Beispiel in ein Artefakt-Repository oder in eine Produktionsumgebung bereitgestellt werden.“	Artefakt-Repository und Produktionsumgebung	Experte 1	15 ff.
„Mit diesen Artefakts [im Artefakt-Repository] kann man dann auch noch sehr gut Rollbacks ausführen.“	Artefakt-Repository	Experte 4	48 ff.
„Kleinen entwickelten Komponenten können mit Versionierung in das Artefakt-Repository bereitgestellt werden. Andere Entwickler können diese Komponente dann aus dem Artefakt-Repository herausziehen und für eigenen Entwicklungen wiederverwenden“	Komponentenwiederverwendung im Artefakt-Repository	Experte 1	37 ff.

Test

Aussage	Kodierung	Experte	Zeilennummer
---------	-----------	---------	--------------

„Typischerweise beginnt es mit der Build-Stage, bei welcher Unit-Tests ausgeführt werden. Für CAP werden dabei die Frameworks Mocha oder Jest verwendet.“	Unit-Tests mit SAP CAP	Experte 1	42 ff.
„Die Integration-Tests werden mit Newman gemacht.“	Integration-Tests mit SAP CAP	Experte 1	46 ff.
„I.d.R. wird in der SAP Q-Unit für [Unit-Tests] verwendet“	Unit-Tests mit SAP UI5	Experte 4	62 ff.
„Für Integration-Tests wird OPA5 verwendet.“	Integration-Tests mit SAP UI5	Experte 4	63 ff.
„Und für System-Tests wird dann i.d.R. WDI5 verwendet.“	System-Tests mit SAP UI5	Experte 4	64 ff.
„Die Empfehlung ist möglichst viel auf den unteren Ebenen abzudecken, also mit Unit-Tests und auf den oberen Ebenen nur noch das zu testen, was man nicht mit Unit- und Integration-Tests testen kann.“	Test-Pyramide	Experte 4	72 ff.

Code-Analysen

Aussage	Kodierung	Experte	Zeilennummer
---------	-----------	---------	--------------

„[Mit SonarQube] wird dann z.B. geprüft, ob ich irgendwelche Lizenzrechte verletze.“	SonarQube	Experte 1	48 ff.
--	-----------	-----------	--------

Vorteile von kontinuierlicher Bereitstellung

Aussage	Kodierung	Experte	Zeilennummer
„Wenn ich dann schnell in eine Canary-Umgebung deploye, kann ich natürlich früh Fehler finden, was schließlich auch deutlich günstiger ist.“	Frühe Fehlerfindung	Experte 1	22 ff.
„So ist die Gefahr, dass etwas im Produktivsystem kaputtgeht sehr gering.“	Wenig Fehler in der Produktion	Experte 2	25 ff.
„[Beim Feature Toggle] wird eine neue Funktionalität dann hinter einer Flag versteckt. Wenn ein bestimmter Kunde dieses Feature dann haben möchte, dann setzt er entsprechend die Flag.“	Feature Toggle	Experte 4	37 ff.

CI/CD-Pipeline-Tools bei der SAP

Aussage	Kodierung	Experte	Zeilennummer
„Zum einen wird der von der SAP bereitgestellte CI/CD-Service verwendet.“	SAP BTP CI/CD	Experte 1	54 ff.

„Des Weiteren gibt es Jenkins. Diese wird i.d.R. mit Project Piper verwendet.“	Jenkins	Experte 1	55 ff.
„Häufig wird für interne Projekte auch Azure DevOps verwendet.“	Azure Pipelines	Experte 1	57 ff.

Aspekte für Wahl einer CI/CD-Pipeline

Aussage	Kodierung	Experte	Zeilennummer
„Für Abteilungen welche keine DevOps-Spezialisten haben spielt die Benutzerfreundlichkeit eine große Rolle. Da ist es zum einen wichtig, wie leicht sich die Tools warten lassen, aber auch wie leicht sich eine Pipeline implementieren lässt. “	Intuitive Bedienbarkeit und Installation und Wartung	Experte 1	63 ff.
„Weiterhin ist wichtig zu wissen, wie flexibel man bei der Pipeline-Gestaltung sein will.“	Flexibilität	Experte 1	65 ff.
„Zudem muss natürlich auch evaluiert werden, welche Funktionalitäten also Tests, Code-Scans und Builds auf der Pipeline ausgeführt werden sollen. “	Tests, Code-Analysen Build	Experte 1	67 ff.

„Zuletzt sollte dann was die Funktionalität angeht auch noch evaluiert werden auf welcher Plattform die Software bereitgestellt werden soll.“	Deploy und Release	Experte 1	68 ff.
„Skalierbarkeit spielt dann auch noch eine wichtige Rolle.“	Skalierbarkeit	Experte 1	69 ff.
„Integration ist auch ein sehr wichtiger Aspekt bei der Auswahl von CI/CD-Pipelines.“	Integrationsmöglichkeiten	Experte 1	76 ff.
„Da muss natürlich auf die Integrationsmöglichkeiten mit dem Repository geachtet werden.“	Integrationsmöglichkeiten von Repositorys	Experte 1	77 ff.
„Sehr selten wird eine CI/CD-Pipeline auch in die Entwicklungsumgebung integriert.“	Integrationsmöglichkeiten von Entwicklungsumgebung	Experte 1	81 ff.
„Was auch aber eher selten in Kundenprojekten beachtet wird, ist die Integrierbarkeit in Projektmanagement-Tools wie Jira.“	Integration in Planungstools	Experte 4	89 ff.
„Wenn du in sehr großen Entwicklungen bist, dann ist es natürlich auch sehr wichtig, dass die Pipeline eine gute Performance besitzt.“	Performance	Experte 2	36 ff.

„Des Weiteren ist es essenziell, dass die CI/CD-Prozesse überwacht werden können.“	Monitoring	Experte 2	37 ff.
„Insbesondere für kleinere Kunden ist es natürlich auch essenziell, wie viel die Pipeline kostet.“	Kosten	Experte 2	43 ff.
„Für viele Kunden ist darüber hinaus der Support wichtig. Es sollten dabei kontinuierliche Updates, Schulungsmaterial sowie wie eine gute Dokumentation verfügbar sein.“	Administrativer Support	Experte 2	44 ff.
„Außerdem ist es für Entwickler immer vorteilhaft, wenn für die Tools eine große Community existiert.“	Administrativer Support	Experte 2	45 ff.
„Für unsere Kunden spielt auch die Sicherheit eine wichtige Rolle.“	Sicherheit	Experte 1	75 ff.

SAP BTP CI/CD-Service

Aussage	Kodierung	Experte	Zeilennummer
„Was bisher noch nicht wirklich funktioniert sind API-Tests.“	Keine API-Tests	Experte 1	72 ff.

„Der SAP CI/CD-Service unterstützt dabei einen ganz normalen Git-Server. Was auch noch funktioniert sind BitBucket Repositorys.“	Unterstützung von Repositorys	Experte 1	78 ff.
„Es können dabei jedoch ausschließlich Commit Events verarbeitet.“	Unterstützung von Commit-Events	Experte 1	80 ff.
„Aber ein direktes Monitoring der Pipeline gibt es nicht.“	Kein Monitoring für SAP CI/CD	Experte 1	82 ff.
„Eine Build-Hour kostet einen Euro.“	Kosten	Experte 1	92 ff.
„Wir können sowohl auf Cloud Foundry als auch auf Kyma deployen.“	Deployment	Experte 1	85 ff.
„Nein leider nicht. Aber die Pipeline kann Software auf das Transport Management System bereitstellen.“	Parallel Build und SAP CTM	Experte 1	95 ff.
„Für interne Projekte darf der SAP BTP CI/CD aufgrund der derzeitigen Produktstandards nicht verwendet werden.“	Nicht für interne Projekte	Experte 2	46 ff.
„Das SAP CI/CD-Tool lohnt sich insbesondere für Kunden, die noch nicht viel DevOps-Expertise besitzen und auch keine teure Infrastruktur betreiben wollen.“	Für Kunden mit wenig Expertise	Experte 2	48 ff.

Azure Pipelines

Aussage	Kodierung	Experte	Zeilennummer
„Hierbei gibt es jetzt auch einige speziellen Governace-Checks, die darüber möglich sind.“	Governance-Checks	Experte 4	52 ff.
„Das SAP Tools Team hatte dann alles auf einen zentralen Blick und konnte entsprechend sagen, dass wenn für eine Pipeline mehr Kapazität benötigt wird, dass entsprechend Ressourcen zugeschaltet werden.“	Erhöhte Flexibilität	Experte 4	53 ff.

Security

Aussage	Kodierung	Experte	Zeilennummer
„Früher gab es dabei immer ein Security-Experten der sich vor der Auslieferung dann um alles kümmern musste.“	Security damals	Experte 3	8 ff.
„Security sollte nicht mehr nur von einem Spezialisten behandelt werden. Vielmehr sollte dies als Kollektiv abgehandelt werden.“	Security heute	Experte 3	32 ff.

„Jeder muss bei der Entwicklung seiner Funktionalitäten schon so früh wie möglich schauen, ob alle sicherheitsrelevanten Aspekte eingehalten wurden. Das wird dann i.d.R. durch Automatisierung gemacht.“	Automatisierung mit Tools	Experte 3	32 ff.
„[Bei statischen Code-Analysen wird insbesondere OS-Scanning betrieben.]“	Automatisierung mit Tools	Experte 3	23 ff.
„[Bei Dynamic Application Security Testing] werden dann auch tatsächlich UI-Elemente, APIs sowie Datenbanken gescannt.“	Dynamic Application Security Testing	Experte 3	24 ff.
„Für CAP Node wird von der SAP das Tool Checkmarx vorgeschrieben. Für Open-Source ist das Tool Whitesource vorgeschrieben.“	Security-Scans für SAP CAP Node	Experte 3	36 ff.

C.6 Expertengewichtung 1

Interviewpartner: Software Architekt SAP DTS Integration (Experte 5)

Datum: 27.03.2023

Interview-Medium: Microsoft-Teams

Funktionalität	Integrationsmöglichkeiten	Kosten	Skalierbarkeit	Performance	Flexibilität	Support	Sicherheit	Benutzerfreundlichkeit	Lokale Gewichtung
Funktionalität	1		2	2	1	2	2	2	0,1852
Integrationsmöglichkeiten	1	1		2	1	2	1	2	0,1605
Kosten	0	1	1	2	2	2	1	2	0,1481
Skalierbarkeit	0	0	1	2	1	2	1	2	0,1235
Performance	0	0	0	1	1	0	0	1	0,0370
Flexibilität	1	1	0	1	1	1	1	2	0,1111
Support	0	0	0	0	2	1	1	2	0,0864
Sicherheit	0	1	1	2	1	1	1	2	0,1235
Benutzerfreundlichkeit	0	0	0	1	0	0	0	1	0,0247
									1,000

Funktionalität	Tests	Build	Deploy/Release	Monitoring	Code-Analysen	Lokale Gewichtung
Tests	1	1	1	2	1	0,2400
Build	1	1	2	2	2	0,3200
Deploy/Release	1	0	1	2	2	0,2400
Monitoring	0	0	0	1	0	0,0400
Code-Analysen	1	0	0	2	1	0,1600
						1,0000

Integrationsmöglichkeiten	Entwicklungsumgebung	Repository	Planungssoftware	Lokale Gewichtung
Entwicklungsumgebung	1	0	2	0,3333
Repository	2	1	2	0,5556
Planungssoftware	0	0	1	0,1111
				1,000

Endfaktoren	
Kriterien	Globale Gewichtung
Test	0,0444
Code-Analysen	0,0296
Build	0,0593
Deploy und Release	0,0444
Monitoring	0,0074
Integration in Repository	0,0892
Integration in Entwicklungsumgebung	0,0535
Integration in Planungssoftware	0,0178
Kosten	0,1481
Skalierbarkeit	0,1235
Integration-Zeit	0,0185
Delivery-Zeit	0,0185
Flexibilität	0,1111
Administrativer Support	0,0648
Community-Support	0,0216
Sicherheit	0,1235
Installation und Wartung	0,0123
Intuitive Bedienbarkeit	0,0123
	1,0000

1 **Interviewer:** Für dich besonders wichtig ist die Funktionalität. Kannst du das
2 begründen?

3 **Experte:** Wenn ich eine Pipeline einbaue, dann ist für mich besonders wichtig, dass
4 die Pipeline bestimmte Funktionalitäten, wie z.B. Tests abdecken. Weniger gewich-
5 tig ist da z.B. die Benutzerfreundlichkeit. Ich bin der Meinung, dass ich eine Pipeline
6 einmal einrichte. Da spielt es dann auch weniger die Rolle wie viel Aufwand das beim
7 initialen Einrichten war.

8 **Interviewer:** Von den Subkriterien der Funktionalität war für dich Build Funktio-
9 nalität besonders wichtig. Kannst du das begründen?

10 **Experte:** Bei Cloud-Native-Projekten werden i.d.R. verschiedene Sprachen und ver-
11 schiedene Frameworks verwendet. Die Pipeline sollte da einfach alle Build-Packages
12 unterstützen. So muss ich dann nicht hergehen und manuell irgendwelche Deploy-
13 ments ausführen. Tests waren für mich auch sehr wichtig. Es ist wichtig, dass ich
14 die Tests mache, bevor es zu Merges kommt. Natürlich kann ich auch Tests abseits
15 von einer CI/CD-Pipeline ausführen. Bei einem größeren Team von Entwickler ist
16 das jedoch kaum noch kontrollierbar.

17 **Interviewer:** Kommen wir zu den Integrationsmöglichkeiten. Deiner Meinung nach
18 ist die Integration eines Repositorys sehr wichtig. Woran liegt das?

19 **Experte:** Oft ist es so, dass sich der Kunde auf eine oder zwei CI/CD und ein oder
20 zwei Repositorys einschießt. Gerade, wenn da die Abhängigkeit mit dem Repository
21 besteht sollte das natürlich in die CI/CD-Pipeline integrierbar sein.

22 **Interviewer:** Kannst du begründen, warum für dich die Integration- und Delivery-
23 Zeit gleich wichtig ist.

24 **Experte:** Aus meiner Sicht gibt es da kein wichtigeres Kriterium, da sowohl CI als
25 auch CD im Hintergrund läuft. Somit ist es für mich jetzt nicht so wichtig, falls eine
26 der beiden Pipelines länger durchläuft.

27 **Interviewer:** Warum ist für dich der administrative Support wichtiger als der
28 Community-Support?

29 **Experte:** Für mich ist es eben sehr wichtig, dass es eine gute Dokumentation gibt. So
30 kann ich z.B. bevor ich eine Pipeline installiere schon abschätzen, wie gut bestimmte

31 Aspekte funktioniere und bin da nicht davon abhängig, dass ich durch Community-
32 Foren irgendwelche Workarounds bekomme. Außerdem ist es mir natürlich auch sehr
33 wichtig, dass die Lösung immer auf dem neusten Stand der Technik ist.

34 **Interviewer:** Kannst du noch einmal deine Entscheidung zur Sicherheit begründen?

35 **Experte:** Also es gehört natürlich einfach zur Developer-Experience dazu, wenn
36 man bestimmte Authentifizierung- und Autorisierungskonzepte hat. Dann ist es
37 natürlich aber auch schon wichtig, dass die Pipeline an sich sicher ist. Gerade die
38 Pipeline bietet natürlich eine sehr gute Möglichkeit feindliche Programme in eine
39 Produktionsumgebung einzuführen.

40

C.7 Expertengewichtung 2

Interviewpartner: Full-Stack-Entwickler SAP DTS Integration (Experte 6)

Datum: 21.03.2023

Interview-Medium: Microsoft-Teams

Funktionalität	Integrationsmöglichkeiten	Kosten	Skalierbarkeit	Performance	Flexibilität	Support	Sicherheit	Benutzerfreundlichkeit	Lokale Gewichtung
Funktionalität	1	1	2	1	1	1	1	1	0,1235
Integrationsmöglichkeiten	1	1	2	1	1	0	1	1	0,1235
Kosten	0	0	2	1	0	0	0	0	0,0123
Skalierbarkeit	1	0	1	0	0	0	0	0	0,0494
Performance	1	1	2	1	0	0	0	0	0,1111
Flexibilität	1	1	2	1	1	0	1	0	0,1235
Support	1	2	2	2	2	1	1	1	0,1728
Sicherheit	1	2	2	2	1	1	1	1	0,1358
Benutzerfreundlichkeit	1	1	2	2	1	1	1	1	0,1481
									1,000

Funktionalität	Tests	Build	Deploy/Release	Monitoring	Code-Analysen	Lokale Gewichtung
Tests	1	2	2	2	2	0,3600
Build	0	1	0	0	0	0,0400
Deploy/Release	0	2	1	2	0	0,2000
Monitoring	0	2	0	1	2	0,2000
Code-Analysen	0	2	2	0	1	0,2000
						1,0000

Integrationsmöglichkeiten	Entwicklungsumgebung	Repository	Planungssoftware	Lokale Gewichtung
Entwicklungsumgebung	1	0	2	0,3333
Repository	2	1	2	0,5556
Planungssoftware	0	0	1	0,1111
				1,000

Endfaktoren	
Kriterien	Globale Gewichtung
Test	0,0444
Code-Analysen	0,0247
Build	0,0049
Deploy und Release	0,0247
Monitoring	0,0247
Integration in Repository	0,0686
Integration in Entwicklungsumgebung	0,0412
Integration in Planungssoftware	0,0137
Kosten	0,0123
Skalierbarkeit	0,0494
Integration-Zeit	0,0833
Delivery-Zeit	0,0278
Flexibilität	0,1235
Administrativer Support	0,0432
Community-Support	0,1296
Sicherheit	0,1358
Installation und Wartung	0,0741
Intuitive Bedienbarkeit	0,0741
	1,0000

1 **Interviewer:** In Tabelle 1 ist für dich besonders wichtig der Support und weni-
2 ger wichtig sind für dich die Kosten. Warum?

3 **Experte:** Ich als Entwickler habe die Erfahrung, dass Tool noch so toll sein kann.
4 Wenn keine gute Dokumentation vorhanden ist, dann hilft mir das als Entwickler
5 nicht viel. Die Kosten sind mir aus Entwicklersicht egal. Da sind mir erstmal die
6 anderen Kriterien wichtiger

7 **Interviewer:** In Tabelle 2 waren dir die Tests besonders wichtig. Kannst du das
8 begründen.

9 **Experte:** Es spart mir sehr viel Zeit, wenn Tests automatisiert werden. Für mich
10 gehört das automatisierte Ausführen von Tests auch zu den Hauptzwecken einer
11 CI/CD-Pipeline. So kann ich frühzeitig erkennen, wenn eine Änderung etwas ka-
12 putt macht.

13 **Interviewer:** Kommen wir zu den Integrationsmöglichkeiten. Warum ist dir die In-
14 tegration in Repositorys besonders wichtig?

15 **Experte:** Ich sehe, dass als eine sehr grundlegende Funktion. Wenn meine CI/CD-
16 Pipeline nicht mit dem Repository integrierbar ist, dann wird das gesamte Konzept
17 einer Pipeline hinfällig. Die Planungssoftware war mir hingegen kaum wichtig, da
18 ich als Entwickler mit so etwas kaum arbeite.

19 **Interviewer:** Kannst du deine Entscheidungen zur Performance begründen?

20 **Experte:** Für mich ist die Integration-Zeit deutlich wichtiger, einfach um da eine
21 höhere Developer-Experience zu bekommen. Wenn ich ein Pull-Request aufmache,
22 dann möchte ich auch schnell Feedback bekommen, ob alles passt oder eben nicht.

23 **Interviewer:** Kannst du deine Entscheidungen zum Support begründen?

24 **Experte:** Ich finde den Community-Support am wichtigsten. Dabei bekommt man
25 den leichtesten Zugriff als Entwickler auf neues Wissen. Weniger geeignet fände ich,
26 wenn ich jedes Mal mit jemanden telefonieren müsste oder immer ein neues Ticket
27 aufmachen müsste. Weiterhin ist es sehr gut, wenn von einer Community Plug-
28 ins bereitgestellt werden. Damit kann ich den Funktionsumfang erweitern. Jedoch
29 besteht hier immer die Gefahr, dass Plug-ins Sicherheitslücken besitzen oder irgend-
30 wann nicht mehr richtig gewartet werden können.

31 **Interviewer:** Wie sieht es mit dem Punkt der Sicherheit aus?

32 **Experte:** Ich finde das Authentifizierungs- und Autorisierungs-Konzept sehr wich-
33 tig, da dies mit der Developer-Experience zusammenhängt. Damit komme ich eben
34 in meinem Entwickleralltag am meisten in Berührung. Wenn die Plattform bspw.
35 SSO enabled ist, dann ist das für mich als Programmierer sehr gemütlich.

36 **Interviewer:** Kannst du deine Entscheidung zur Benutzerfreundlichkeit begründen?

37 **Experte:** Sowohl mit Wartung bzw. Installation und mit der gewöhnlichen Bedien-
38 barkeit kommt man gelegentlich in Berührung. Das sollte deshalb beides einigerma-
39 ßen gut funktionieren.

40

C.8 Expertengewichtung 3

Interviewpartner: Test Developer SAP Hyperspace Adoption & Onboarding (Experte 4)

Datum: 22.03.2023

Interview-Medium: Microsoft-Teams

	Funktionalität	Integrationsmöglichkeiten	Kosten	Skalierbarkeit	Performance	Flexibilität	Support	Sicherheit	Benutzerfreundlichkeit	Lokale Gewichtung
Funktionalität	1	2	2	2	2	2	2	0	2	0,1852
Integrationsmöglichkeiten	0	1	1	1	2	1	2	0	2	0,1358
Kosten	0	0	1	0	2	2	1	0	1	0,0741
Skalierbarkeit	0	0	1	1	1	1	2	0	2	0,1358
Performance	0	0	0	1	1	1	2	0	2	0,0988
Flexibilität	0	1	1	0	0	1	1	0	1	0,0617
Support	0	0	0	0	0	0	1	1	0	0,0370
Sicherheit	2	2	2	2	2	2	2	1	2	0,2099
Benutzerfreundlichkeit	0	0	0	0	0	1	2	0	1	0,0617
										1,000

Funktionalität	Tests	Build	Deploy/Release	Monitoring	Code-Analysen	Lokale Gewichtung
Tests	1	1	2	2	1	0,2800
Build	1	1	2	2	1	0,2800
Deploy/Release	0	0	1	1	1	0,1200
Monitoring	0	0	1	1	1	0,1200
Code-Analysen	1	1	1	1	1	0,2000
						1,0000

Integrationsmöglichkeiten	Entwicklungsumgebung	Repository	Planungssoftware	Lokale Gewichtung
Entwicklungsumgebung	1	0	2	0,3333
Repository	2	1	2	0,5556
Planungssoftware	0	0	1	0,1111
				1,000

Performance				
	Integration-Zeit		Delivery-Zeit	Lokale Gewichtung
	1	2		0,7500
	0	1		0,2500
				1,000

Support				
	Administrativer Support	Community-Support		Lokale Gewichtung
	1	0		0,2500
	2	1		0,7500
				1,000

Benutzerfreundlichkeit				
	Installation und Wartung	Intuitive Bedienbarkeit		Lokale Gewichtung
	1	0		0,2500
	2	1		0,7500
				1,0000

Endfaktoren	
Kriterien	Globale Gewichtung
Test	0,0519
Code-Analysen	0,0370
Build	0,0519
Deploy und Release	0,0222
Monitoring	0,0222
Integration in Repository	0,0754
Integration in Entwicklungsumgebung	0,0453
Integration in Planungssoftware	0,0151
Kosten	0,0741
Skalierbarkeit	0,1358
Integration-Zeit	0,0741
Delivery-Zeit	0,0247
Flexibilität	0,0617
Administrativer Support	0,0093
Community-Support	0,0278
Sicherheit	0,2099
Installation und Wartung	0,0154
Intuitive Bedienbarkeit	0,0463
	1,0000

1 **Interviewer:** Besonders wichtig ist für dich die Sicherheit. Kannst du das be-
2 gründen?

3 **Experte:** Sicherheitslücken in unseren Systemen würden einen sehr großen Image-
4 schaden verursachen. Deswegen ist das für mich das absolute Top-Thema. Gerade
5 wenn wir neue Software bereitstellen, darf es nicht passieren, dass eventuell feindli-
6 che Programme eingeschleust werden. Nicht so wichtig ist für mich der Support, da
7 dieser ganz schnell zum Bottleneck werden kann. Wichtiger ist da für mich z.B. die
8 Benutzerfreundlichkeit. Man sollte die Nutzer gar nicht erst in die Situation kom-
9 men lassen, dass diese Support benötigen.

10 **Interviewer:** Warum ist für dich der administrative Support wichtiger als der
11 Community-Support?

12 **Experte:** Es ist viel wichtiger, dass man eine Community besitzt, welche Fragen
13 beantworten kann. Dies ist eigentlich die einzige Möglichkeit sowas skalierbar zu ma-
14 chen. Man wird niemals ein Support-Team besitzen, was groß genug ist alle Fragen
15 zu beantworten. Man könnte den administrativen Support also nicht gut skalieren.

16 **Interviewer:** Im Kriterium der Funktionalität ist für dich besonders wichtig die
17 Test- und Build-Funktionalität. Warum?

18 **Experte:** Test ist natürlich mein Hintergrund, da ich von dieser Seite her komme.
19 Es ist für mich einfach wichtig, dass ich etwas ausliefere, was auch validiert ist.

20 **Interviewer:** Warum ist für dich die Integration-Zeit wichtiger als die Delivery-
21 Zeit?

22 **Experte:** Es kann eben aus Entwicklersicht nicht sein, dass ich eine Änderung im
23 Code mache und dann erst einmal eine halbe Stunde warten muss, bis ich Feedback
24 bekomme. Da geht die Motivation im Team verloren. Und es stapeln sich einfach
25 die Änderungen, bevor ich die Änderungen dann tatsächlich ausliefere.

26 **Interviewer:** Kannst du noch einmal deine Entscheidung begründen?

27 **Experte:** Authentifizierung ist natürlich wichtig, damit nicht jemand unberechtig-
28 tes ein Deployment durchführen kann, welcher das eigentlich gar nicht sollte.

29 **Interviewer:** Für dich war die Installation und Wartung weniger wichtig als die
30 intuitive Bedienbarkeit?

³¹ **Experte:** Installation und Wartung betrifft mich einmal beim Setup, während die
³² intuitive Bedienbarkeit kontinuierlich wichtig ist.

³³

C.9 Expertengewichtung 4

Interviewpartner: Backend Test Developer SAP DTS Integration (Experte 7)

Datum: 22.03.2023

Interview-Medium: Microsoft-Teams

	Funktionalität	Integrationsmöglichkeiten	Kosten	Skalierbarkeit	Performance	Flexibilität	Support	Sicherheit	Benutzerfreundlichkeit	Lokale Gewichtung
Funktionalität	1	1	2	1	1	1	1	1	1	0.1235
Integrationsmöglichkeiten	1	1	1	2	1	1	0	1	1	0.1235
Kosten	0	0	1	0	0	0	0	0	0	0.0123
Skalierbarkeit	1	0	2	1	0	0	0	0	0	0.0494
Performance	1	1	2	2	1	1	0	1	0	0.1111
Flexibilität	1	1	2	2	1	0	1	1	1	0.1235
Support	1	2	2	2	2	2	1	1	1	0.1728
Sicherheit	1	1	2	2	1	1	1	1	1	0.1358
Benutzerfreundlichkeit	1	1	2	2	2	1	1	1	1	0.1481
										1,000

	Funktionalität	Integrationsmöglichkeiten	Kosten	Skalierbarkeit	Performance	Flexibilität	Support	Sicherheit	Benutzerfreundlichkeit	Lokale Gewichtung
Funktionalität	1	1	2	1	1	1	1	1	1	0.3600
Integrationsmöglichkeiten	1	1	1	2	1	1	0	1	1	0.2000
Kosten	0	0	1	0	0	0	0	0	0	0.2800
Skalierbarkeit	1	0	2	1	0	0	0	0	0	0.1200
Performance	1	1	2	2	1	0	1	1	1	0.0400
Flexibilität	1	1	2	2	1	0	1	1	1	1,0000
Support	1	2	2	2	2	2	1	1	1	
Sicherheit	1	1	2	2	1	1	1	1	1	
Benutzerfreundlichkeit	1	1	2	2	2	1	1	1	1	

	Funktionalität	Integrationsmöglichkeiten	Kosten	Skalierbarkeit	Performance	Flexibilität	Support	Sicherheit	Benutzerfreundlichkeit	Lokale Gewichtung
Funktionalität	1	1	2	1	1	1	1	1	1	0.2222
Integrationsmöglichkeiten	1	1	1	2	1	1	0	1	1	0.4444
Kosten	0	0	1	0	0	0	0	0	0	0.3333
Skalierbarkeit	1	0	2	1	0	0	0	0	0	1,000
Performance	1	1	2	2	1	0	1	1	1	
Flexibilität	1	1	2	2	1	0	1	1	1	
Support	1	2	2	2	2	2	1	1	1	
Sicherheit	1	1	2	2	1	1	1	1	1	
Benutzerfreundlichkeit	1	1	2	2	2	1	1	1	1	

Performance					
		Integration-Zeit		Delivery-Zeit	Lokale Gewichtung
Integration-Time		1	2		0,7500
Delivery-Time		0	1		0,2500
					1,000

Support					
		Administrativer Support	Community-Support		Lokale Gewichtung
Administrativer Support		1	0		0,2500
Community-Support		2	1		0,7500
					1,000

Benutzerfreundlichkeit					
		Installation und Wartung	Intuitive Bedienbarkeit		Lokale Gewichtung
Installation und Wartung		1	0		0,2500
Intuitive Bedienbarkeit		2	1		0,7500
					1,0000

Endfaktoren	
Kriterien	Globale Gewichtung
Test	0,0711
Code-Analysen	0,0079
Build	0,0395
Deploy und Release	0,0553
Monitoring	0,0237
Integration in Repository	0,0823
Integration in Entwicklungsumgebung	0,0412
Integration in Planungssoftware	0,0617
Kosten	0,1235
Skalierbarkeit	0,1358
Integration-Zeit	0,0556
Delivery-Zeit	0,0185
Flexibilität	0,1605
Administrativer Support	0,0123
Community-Support	0,0370
Sicherheit	0,0617
Installation und Wartung	0,0031
Intuitive Bedienbarkeit	0,0093
	1,0000

1 **Interviewer:** Fangen wir mit dem Subkriterium Funktionalität an. Kannst du
2 deine Entscheidungen bitte begründen.

3 **Experte:** Also für mich sind fast alle Funktionalitäten gleichgewichtig. Ich benötige
4 alle diese Funktionalitäten, weil meine CI/CD-Pipeline sonst nicht sonderlich nützlich
5 ist. Zu den Tests, der Hauptgrund der CI/CD-Pipeline ist es eigentlich Tests zu au-
6 tomatisieren. Ohne Build, Deploy und Release funktioniert meine Pipeline nicht.
7 Ohne Monitoring kann ich nicht evaluieren, ob etwas fehlschlägt oder nicht. Code-
8 Analysen sind hingegen bei Kundenprojekten oft nicht verpflichtend.

9 **Interviewer:** Machen wir mit den Integrationsmöglichkeiten weiter. Für dich wa-
10 ren sowohl die Integration in das Repository als auch in Planungssoftware wichtig.
11 Warum?

12 **Experte:** Ich benötige auf jeden Fall mein Code für die Pipeline. Deswegen ist die
13 Integration in das Repository eine essenzielle Funktionalität. Planungssoftware ist
14 auch sehr wichtig, da das Business nicht in den Code reinschaut, sondern in eine
15 Planungssoftware. In vielen Projekten, in den ich war werden, die Ergebnisse der
16 Code-Analysen unmittelbar in der Planungssoftware angezeigt.

17 **Interviewer:** Für dich war die Integration-Zeit wichtiger als die Delivery-Zeit.
18 Warum?

19 **Experte:** Die Integration-Zeit ist da einfach wichtig, um einen schnellen Feedback-
20 Zyklus zu haben. Außerdem habe ich bereits in vielen Projekten gearbeitet, die ein
21 Blue-Green-Deployment verwendet haben. So konnte nach Validierung lediglich auf
22 eine neue Version umgeschaltet werden, wobei der Entwickler nicht durchgehend den
23 Delivery-Prozess beobachten muss.

24 **Interviewer:** Warum ist für dich sowohl der Community-Support als auch der ad-
25 ministrative Support gleichgewichtig?

26 **Experte:** Natürlich ist Community-Support sehr wichtig. Andererseits ist es so, dass
27 es natürlich auch sehr wichtig eine gute Doku und Schulungsunterlagen zu besitzen.

28 **Interviewer:** Kannst du deine Entscheidung zur Benutzerfreundlichkeit begründen?

29 **Experte:** Eigentlich ist es so, dass man ein Pipeline-System einmal aufsetzt und
30 dann nicht mehr. Was i.d.R. häufiger gemacht wird insbesondere in einer Composable-

31 Enterprise-Architektur ist das Aufsetzen von Pipelines.

32 **Interviewer:** Noch einmal zu den Kriterien auf oberster Ebene. Warum ist dir die
33 Sicherheit und die Funktionalität so wichtig?

34 **Experte:** Das Bereitstellen von Software schöpft Wert für das Unternehmen. Des-
35 wegen ist es einfach wichtig, dass keine in meine Pipelines eingreifen kann und diesen
36 Prozess stören kann.

37

C.10 Expertengewichtung 5

Interviewpartner: Product Management CLM (Experte 8)

Datum: 22.03.2023

Interview-Medium: Microsoft-Teams

	Funktionalität	Integrationsmöglichkeiten	Kosten	Skalierbarkeit	Performance	Flexibilität	Support	Sicherheit	Benutzerfreundlichkeit	Lokale Gewichtung
Funktionalität	1	1	2	1	1	1	1	1	1	0.1235
Integrationsmöglichkeiten	1	1	1	2	1	1	0	1	1	0.1235
Kosten	0	0	1	0	0	0	0	0	0	0.0123
Skalierbarkeit	1	0	2	1	0	0	0	0	0	0.0494
Performance	1	1	2	2	1	1	0	1	0	0.1111
Flexibilität	1	1	2	2	1	1	0	1	1	0.1235
Support	1	2	2	2	2	2	1	1	1	0.1728
Sicherheit	1	1	2	2	1	1	1	1	1	0.1358
Benutzerfreundlichkeit	1	1	2	2	2	1	1	1	1	0.1481
										1,000

Funktionalität	Tests	Build	Deploy/Release	Monitoring	Code-Analysen	Lokale Gewichtung
Tests	1	2	2	2	2	0.3600
Build	0	1	0	0	2	0.2000
Deploy/Release	0	2	1	1	2	0.2800
Monitoring	0	0	0	1	2	0.1200
Code-Analysen	0	0	0	0	1	0.0400
						1,000

Integrationsmöglichkeiten	Entwicklungsumgebung	Repository	Planungssoftware	Lokale Gewichtung
Entwicklungsumgebung	1	0	1	0.2222
Repository	2	1	1	0.4444
Planungssoftware	1	1	1	0.3333
				1,000

Performance					
		Integration-Zeit		Delivery-Zeit	Lokale Gewichtung
Integration-Time		1	2		0,7500
Delivery-Time		0	1		0,2500
					1,000

Support					
		Administrativer Support		Community-Support	Lokale Gewichtung
Administrativer Support		1	0		0,2500
Community-Support		2	1		0,7500
					1,000

Benutzerfreundlichkeit					
		Installation und Wartung		Intuitive Bedienbarkeit	Lokale Gewichtung
Installation und Wartung		1	0		0,2500
Intuitive Bedienbarkeit		2	1		0,7500
					1,0000

Endfaktoren	
Kriterien	Globale Gewichtung
Test	0,0711
Code-Analysen	0,0079
Build	0,0395
Deploy und Release	0,0553
Monitoring	0,0237
Integration in Repository	0,0823
Integration in Entwicklungsumgebung	0,0412
Integration in Planungssoftware	0,0617
Kosten	0,1235
Skalierbarkeit	0,1358
Integration-Zeit	0,0556
Delivery-Zeit	0,0185
Flexibilität	0,1605
Administrativer Support	0,0123
Community-Support	0,0370
Sicherheit	0,0617
Installation und Wartung	0,0031
Intuitive Bedienbarkeit	0,0093
	1,0000

1 **Interviewer:** Fangen wir auf oberster Ebene an. Besonders wichtig war für die
2 Integration. Warum?

3 **Experte:** Wir haben bei Kunden das Gefühl, dass diese möchten, dass die CI/CD-
4 Tools in ihre Prozesse integrierbar sind. Da spielt es natürlich insbesondere eine
5 sehr große Rolle, wo diese ihr Repository haben. Sicherheit kann dann natürlich
6 auch ein K.O.-Kriterium sein. Wenn da keine angemessenen getroffen werden, kann
7 das natürlich ein Grund sein eine Pipeline nicht zu wählen.

8 **Interviewer:** Machen wir mit dem Subkriterium Funktionalität weiter.

9 **Experte:** Der Build war für mich sehr wichtig. Das liegt einfach daran, dass man
10 ohne den Build gar nicht erst weiter kommt. Natürlich ist der Hauptgrund von
11 CI/CD-Pipelines automatisierte Tests durchzuführen, jedoch funktioniert das ohne
12 den Build erst gar nicht. Es gibt einige Kunden die auch ganz stark auf Compliance
13 achten, weswegen das schon auch gleichwertig wie die Test-Funktionalität ist. Was
14 Deploy und Release angeht, es gibt auch einige Kunden die auch Sachen, dass sie
15 darauf verzichten.

16 **Interviewer:** Kommen wir zur Integration. Für dich ist sehr wichtig die Integration
17 in das Repository. Warum?

18 **Experte:** Ich denke, dass das Repository ziemlich eindeutig ist. Das gehört mei-
19 ner Meinung nach zur Voraussetzung. Was die Planungssoftware angeht, haben die
20 Kunden meistens isolierte Lösungen.

21 **Interviewer:** In der Kategorie der Performance war dir die Integration-Zeit wich-
22 tiger. Warum?

23 **Experte:** Meiner Erfahrung nach war es für die Kunden oft in Ordnung, wenn die
24 Delivery-Zeit ein wenig länger dauert. Das liegt auch daran, dass vor dem Deploy
25 noch oft manuelle Schritte gemacht werden.

26 **Interviewer:** Kannst du deine Entscheidung zum Support begründen?

27 **Experte:** Ich habe den administrativen Support jetzt höher gewertet, da ich die
28 Erfahrung gemacht habe, dass Kunden sehr viel Wert auf die SLAs legen. So weiß
29 der Kunde natürlich genau, dass er sich auch am Wochenende melden kann, wenn
30 er ein Problem hat und dann auch entsprechend eine Antwort bekommt.

31 **Interviewer:** Kannst du auch noch deine Entscheidung zur Benutzerfreundlichkeit
32 begründen.

33 **Experte:** Ich habe die Intuitive Bedienbarkeit und Installation und Wartung auf
34 eine Wichtigkeitsstufe gesetzt. Zum einen ist es natürlich so, dass die intuitive Be-
35 dienbarkeit etwas ist, mit welchem man alltäglich zu tun hat. Aber gerade bezüglich
36 des Wartens von komplexen Infrastrukturen, war es für viele Kunden der Grund sich
37 dann letztendlich für eine SaaS-Lösung zu entscheiden.

38

C.11 Expertengewichtung Durchschnitt

	Expertengewichtung 1	Expertengewichtung 2	Expertengewichtung 3	Expertengewichtung 4	Expertengewichtung 5	Durchschnitt
Funktionalität	0,1852	0,1235	0,1852	0,1975	0,1728	0,1728
Integrationsmöglichkeiten	0,1605	0,1235	0,1358	0,1852	0,1852	0,1580
Kosten	0,1481	0,0123	0,0741	0,1235	0,1235	0,0963
Skalierbarkeit	0,1235	0,0494	0,1358	0,1358	0,1358	0,1160
Performance	0,0370	0,1111	0,0988	0,0741	0,0741	0,0790
Flexibilität	0,1111	0,1235	0,0617	0,1605	0,1358	0,1185
Support	0,0864	0,1728	0,0370	0,0494	0,0617	0,0815
Sicherheit	0,1235	0,1358	0,2099	0,0617	0,0988	0,1259
Benutzerfreundlichkeit	0,0247	0,1481	0,0617	0,0123	0,0123	0,0519
						1

Funktionalität	Expertengewichtung 1	Expertengewichtung 2	Expertengewichtung 3	Expertengewichtung 4	Expertengewichtung 5	Durchschnitt
Tests	0,2400	0,3600	0,2800	0,3600	0,3600	0,3200
Build	0,3200	0,0400	0,2800	0,2000	0,2000	0,2080
Deploy/Release	0,2400	0,2000	0,1200	0,2800	0,2800	0,2240
Monitoring	0,0400	0,2000	0,1200	0,1200	0,1200	0,1200
Code-Analysen	0,1600	0,2000	0,2000	0,0400	0,0400	0,1280
						1

Integrationsmöglichkeiten	Expertengewichtung 1	Expertengewichtung 2	Expertengewichtung 3	Expertengewichtung 4	Expertengewichtung 5	Durchschnitt
Entwicklungsumgebung	0,3333	0,3333	0,3333	0,2222	0,2222	0,2889
Repository	0,5556	0,5556	0,5556	0,4444	0,4444	0,5111
Planungssoftware	0,1111	0,1111	0,1111	0,3333	0,3333	0,2000
						1

Performance										
	Expertengewichtung 1	Expertengewichtung 2	Expertengewichtung 3	Expertengewichtung 4	Expertengewichtung 5	Durchschnitt				
Integration-Zeit	0,5000	0,7500	0,7500	0,7500	0,7500	0,7000				
Delivery-Zeit	0,5000	0,2500	0,2500	0,2500	0,2500	0,3000				
						1				

Support										
	Expertengewichtung 1	Expertengewichtung 2	Expertengewichtung 3	Expertengewichtung 4	Expertengewichtung 5	Durchschnitt				
Administrativer Support	0,7500	0,2500	0,2500	0,2500	0,2500	0,3500				
Community-Support	0,2500	0,7500	0,7500	0,7500	0,7500	0,6500				
						1				

Benutzerfreundlichkeit										
	Expertengewichtung 1	Expertengewichtung 2	Expertengewichtung 3	Expertengewichtung 4	Expertengewichtung 5	Durchschnitt				
Installation und Wartung	0,5000	0,5000	0,2500	0,2500	0,5000	0,4000				
Intuitive Bedienbarkeit	0,5000	0,5000	0,7500	0,7500	0,5000	0,6000				
						1				

[illegible]