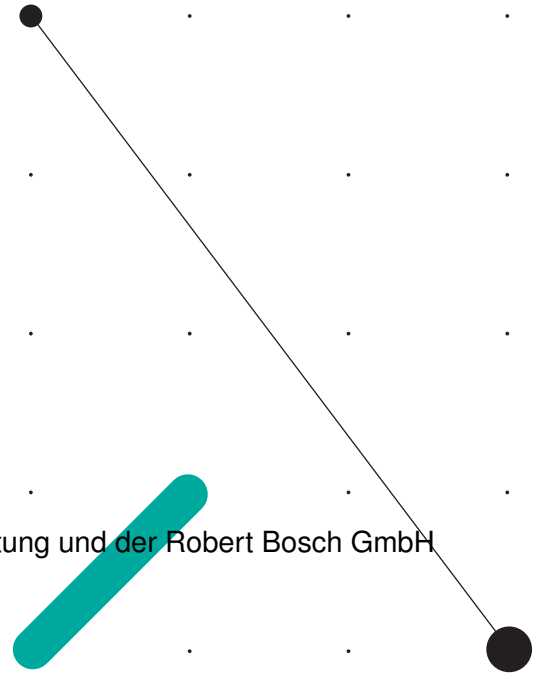
Hochschule Konstanz
Department of Computer Science

Bachelorarbeit

KI-Integration in Cloud Native Platform Engineering: Eine systematische Analyse aktueller Lösungsansätze und deren praktische Anwendung



von

Nils Arnold

zur Erlangung des akademischen Grades

Bachelor of Science

im Studiengang Wirtschaftsinformatik

an der Hochschule Konstanz Technik, Wirtschaft und Gestaltung und der Robert Bosch GmbH

Matrikelnummer:	307179
Abgabedatum:	31.01.2026
Erstbetreuer:	Prof. Dr. Johannes Schneider
Zweitbetreuer:	Lukas Grodmeier.(M.Sc.)

Abstract

Abstract. . .

Ehrenwörtliche Erklärung

Hiermit erkläre ich, Nils Arnold, geboren am 18. Januar 2003 in Balingen,

(1) dass ich meine Bachelorarbeit mit dem Titel:

„KI-Integration in Cloud Native Platform Engineering: Eine systematische Analyse aktueller Lösungsansätze und deren praktische Anwendung“

bei der Robert Bosch GmbH unter Anleitung von Prof. Dr. Johannes Schneider und Lukas Grodmeier (M.Sc.) selbstständig und ohne fremde Hilfe angefertigt habe und keine anderen als die angeführten Hilfen benutzt habe;

(2) dass ich die Übernahme wörtlicher Zitate, von Tabellen, Zeichnungen, Bildern und Programmen aus der Literatur oder anderen Quellen (Internet) sowie die Verwendung der Gedanken anderer Autoren an den entsprechenden Stellen innerhalb der Arbeit gekennzeichnet habe.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Ort, Datum	Unterschrift

Inhaltsverzeichnis

1

Einleitung

Cloud-native Technologien und Platform-Engineering-Ansätze prägen zunehmend den Aufbau und Betrieb moderner Softwareplattformen. In den vergangenen Jahren hat sich ein breites Spektrum kommerzieller Produkte und Open-Source-Lösungen etabliert, das zentrale Aufgaben des Plattformbetriebs adressiert. Diese Entwicklungen verlaufen in hohen Innovationszyklen und führen zu stetig neuen Anforderungen an Architektur und Betrieb.
Parallel dazu gewinnen KI-gestützte Werkzeuge im Plattformumfeld stark an Bedeutung. Sie versprechen, komplexe Betriebsdaten auszuwerten, Abläufe zu automatisieren und Entscheidungen im Plattformbetrieb zu unterstützen. Die Dynamik und Vielfalt dieser Ansätze ist hoch und wächst zum Teil sogar schneller als die reinen Plattformtechnologien selbst.
Marktanalysen zeigen, wie rasant sich diese Felder entwickeln: Laut Fortune Business Insights werden bis 2025 etwa 95 % aller neuen digitalen Workloads auf Cloud-native Plattformen laufen, verglichen mit rund 30 % im Jahr 2021 [**nothbaumCloudnativeErklaertArchitektur**].
Vor dem Hintergrund dieser hohen Innovationsgeschwindigkeit stellt sich die Frage, wie der Einsatz von KI-Technologien im Cloud-Native Platform Engineering sinnvoll eingeordnet und bewertet werden kann.

1.1. Problemstellung

Im Plattformbetrieb treffen Teams heute auf eine schnell wachsende Zahl technischer Optionen. Plattform-Stacks entwickeln sich laufend weiter und werden durch neue Kom-

ponenten, Schnittstellen und Betriebsmodelle erweitert. Branchenberichte beschreiben diese Entwicklung als anhaltenden Trend und zeigen zugleich die zunehmende Breite des Cloud-Native-Ökosystems [**StateCloudNative2025**].
Zusätzlich entstehen in kurzer Folge KI-gestützte Werkzeuge, die im Plattformkontext eingesetzt werden können. Diese Ansätze unterscheiden sich jedoch stark im Reifegrad und erfordern je nach Lösung unterschiedliche Integrations- und Betriebsaufwände. Fachanalysen weisen darauf hin, dass Teams dadurch vermehrt vor schwer vergleichbaren Tool- und Strategieentscheidungen stehen [**InfoQCloudDevOps**].
In der Praxis fehlt Plattform-Teams damit eine belastbare Grundlage, um relevante Technologieoptionen systematisch zu vergleichen. Eine Evaluierung über viele Prototypen ist meist nicht realistisch. Dadurch entstehen Unsicherheit und Reibungsverluste bei Auswahl- und Priorisierungsentscheidungen.

1.2. Zielsetzung der Arbeit

Ziel dieser Arbeit ist es, eine strukturierte Entscheidungsgrundlage für den Einsatz von KI im Cloud-Native Platform Engineering zu erarbeiten. Dafür wird der aktuelle Forschungsstand systematisch erfasst und so aufbereitet, dass typische Anwendungsfelder und passende Lösungsansätze klar voneinander abgegrenzt werden können.
Darauf aufbauend wird ein Bewertungsframework entwickelt, das KI-Use-Cases entlang zentraler Kriterien vergleichbar macht. Im Fokus stehen dabei insbesondere der erwartbare operative Nutzen sowie der Aufwand, der für eine Integration und den laufenden Betrieb erforderlich ist.
Die praktische Anwendbarkeit des Bewertungskonzepts wird anhand der Bosch Digital Manufacturing Plattform geprüft. Ergebnis der Arbeit ist eine nachvollziehbare Einordnung relevanter KI-Anwendungsfelder sowie eine daraus abgeleitete Priorisierung, die als Grundlage für weitere Entscheidungen im Plattformbetrieb dienen kann.

1.3. Aufbau der Arbeit

Kapitel 2 stellt die theoretischen Grundlagen sowie relevante verwandte Arbeiten aus den Bereichen Cloud Native Technologien, Platform Engineering und Künstliche Intelligenz vor. In Kapitel 3 wird das methodische Vorgehen beschrieben, insbesondere die Durchführung der systematischen Mapping Study.
Die Ergebnisse der Literaturanalyse werden in Kapitel 4 dargestellt und in einem kon-

zeptionellen Matching-Framework zusammengeführt. Kapitel 5 überträgt dieses Framework auf die Bosch Digital Manufacturing Plattform und wendet es auf identifizierte Problemfelder an.
Abschließend werden die Ergebnisse in Kapitel 6 diskutiert und in Kapitel 7 zusammengefasst sowie ein Ausblick auf weiterführende Fragestellungen gegeben.

2

Grundlagen und verwandte Arbeiten

In diesem Kapitel werden die theoretischen Grundlagen dargestellt, die für das Verständnis der Arbeit notwendig sind. Hierzu werden zentrale Begriffe aus dem Bereich Cloud Native Platform Engineering, DevOps, CI/CD, Lernparadigmen des maschinellen Lernens und AIOps erläutert. Anschließend werden relevante verwandte Arbeiten beschrieben und kritisch eingeordnet.

2.1. Grundlagen

Ziel dieses Abschnitts ist es, die für die Arbeit relevanten Konzepte aus Cloud-Native-Technologien, DevOps sowie Künstlicher Intelligenz (KI) im Plattformbetrieb strukturiert darzustellen. Die dargestellten Konzepte bilden die Basis für die anschließende Literaturanalyse und die Auswertung der KI-Anwendungen im Platform Engineering.

2.1.1. Cloud Native Technologien und Platform Engineering

Cloud Native (CN) gewann ab 2013 insbesondere durch die Etablierung von Containertechnologien bis hin zu Kubernetes an Bedeutung. Im Kern beschreibt CN heute weniger eine einzelne Technologie, sondern ein Zielbild für modular aufgebaute Systeme (z.B. Microservices), die sich gut bereitstellen, skalieren und resilient betreiben lassen. Die CNCF fasst Cloud Native als Ansatz für skalierbare Anwendungen in dynamischen Cloud-Umgebungen zusammen [1].
Kubernetes hat sich dabei als zentrale Plattform etabliert. Es dient als portable, er-

weiterbare Open-Source-Lösung zur Verwaltung containerisierter Workloads und unterstützt insbesondere deklarative Konfiguration und Automatisierung. Aufbauend auf diesem deklarativen Ansatz wird Kubernetes häufig in GitOps-basierten Betriebsmodellen eingesetzt, bei denen der gewünschte Systemzustand versionskontrolliert abgelegt und automatisiert mit dem laufenden Cluster abgeglichen wird [2, 3].
Im Platform Engineering werden diese Grundlagen genutzt, um Entwicklerteams über eine interne Plattform (Internal Developer Platform) standardisierte, sichere Self-Service-Capabilities bereitzustellen und damit Lieferfähigkeit, Compliance und Betrieb zu verbessern [4].

2.1.2. DevOps und CI/CD

Development und Operations (DevOps) ist ein kultureller und organisatorischer Ansatz, bei dem Entwicklung und Betrieb gemeinsam Verantwortung für den gesamten Software-Lebenszyklus tragen. DevOps entstand vor allem als Antwort auf längere Release-Zyklen und Silos zwischen Entwicklung und Betrieb, die schnelle Änderungen in produktiven Systemen erschwerten. Ziel ist es, Zusammenarbeit und Kommunikation zu stärken und Abläufe so zu gestalten, dass Änderungen häufiger, zuverlässiger und mit klaren Verantwortlichkeiten bereitgestellt werden können. Der DevOps-Ansatz entstand im Kontext verteilter Plattformumgebungen, in denen klassische Trennungen zwischen Entwicklung und Betrieb an ihre Grenzen stießen [5, 6].
Continuous Integration (CI) und Continuous Delivery/Deployment (CD) sind zentrale Praktiken innerhalb von DevOps. CI beschreibt die regelmäßige und automatisierte Integration von Codeänderungen in ein gemeinsames Repository inklusive Build und Tests. CD baut darauf auf und automatisiert die Auslieferung. Continuous Delivery endet vor dem automatischen Produktiv-Deployment, während Continuous Deployment diesen Schritt ebenfalls automatisiert [**WasIstCI**, 7].

2.1.3. Lernparadigmen des maschinellen Lernens

Die grundlegenden Lernparadigmen des maschinellen Lernens lassen sich in Supervised Learning (SL), Unsupervised Learning (UL) und Reinforcement Learning (RL) einteilen.
SL trainiert ein Modell anhand gelabelter Daten, sodass es eine Abbildung von Eingaben auf eine Ziel- bzw. Antwortvariable lernt [8, 9].
UL arbeitet mit ungelabelten Daten und zielt darauf ab, darin Strukturen, Cluster oder Auffälligkeiten zu erkennen, ohne dass eine explizite Zielvariable vorgegeben ist [8, 10].

RL beschreibt Verfahren, bei denen ein Agent durch Interaktion mit einer Umgebung Handlungen lernt, um eine langfristige (kumulative) Belohnung zu maximieren [8, 11]. Die drei Lernparadigmen bilden die Grundlage, während die konkrete Umsetzung in der Praxis typischerweise über spezifische Algorithmen und Methoden erfolgt. Tabelle 2.1 fasst die in dieser Arbeit betrachteten Algorithmusklassen zusammen und schafft damit eine einheitliche begriffliche Grundlage für die spätere Auswertung.

Tabelle 2.1: Beschreibung Algorithmen und Methoden

Algorithmen und Methoden	Beschreibung
Deep Learning (DL)/ Neuronale Netze (NN)	Erfassen komplexe Muster in Daten und eignen sich besonders für unstrukturierte Eingaben wie Log- oder Monitoring-Daten.
Ensemble und Baum-basiert	Kombinieren mehrere Modelle zur Steigerung der Vorhersagegenauigkeit und verarbeiten große sowie semi-strukturierte Datensätze effizient.
Klassische Klassifikation/ Regression	Traditionelle ML-Modelle zur Vorhersage von Mustern oder Ereignissen auf Basis strukturierter Daten.
Clustering	Gruppieren Datenpunkte ohne Labels in inhaltlich ähnliche Cluster und unterstützen dadurch Mustererkennung und Anomaliedetektion.

In der Analyse wurden Lernparadigmen zudem auch dann zugeordnet, wenn sie in den Publikationen nicht explizit benannt waren, sondern nur über typische Aufgabenstellungen erkennbar wurden. Solche Aufgabenstellungen werden in den in diesem Abschnitt zitierten Quellen ausführlicher beschrieben.

2.1.4. AIOps und verwandte Konzepte

AIOps (Artificial Intelligence for IT Operations) bezeichnet den Einsatz von KI-Technologien zur Automatisierung und Optimierung von IT-Betriebsprozessen. Im Kontext dieser Arbeit liegt der Fokus auf ÄI for Ops", also der Anwendung von KI zur Überwachung, Analyse und Optimierung von Cloud-native Plattformen und deren Betrieb. Im Gegensatz zu klassischen Monitoring-Ansätzen, die überwiegend auf statischen Schwellenwerten basieren, ermöglichen KI-gestützte Verfahren eine proaktive Erkennung von Anomalien und betrieblichen Mustern auf Basis großer, heterogener Betriebsdaten. AIOps ist dabei klar von MLOps abzugrenzen, das primär die Entwicklung, das Training und den Lebenszyklus von KI-Modellen adressiert, während AIOps den stabilen und effizienten Plattformbetrieb unterstützt [12, 13].

2.2. Verwandte Arbeiten

Dieser Abschnitt ordnet zentrale Arbeiten zur KI-Integration im Cloud-Native Platform Engineering ein und grenzt den Fokus der Arbeit ab. Im Mittelpunkt steht KI-gestützter Plattformbetrieb (AI for Ops) als operative Unterstützung für Platform Engineers, nicht MLOps als Infrastruktur für datenwissenschaftliche Arbeitsabläufe.
Ein Teil der Literatur untersucht KI zur Automatisierung von Kubernetes- und Cloud-Plattformen. Dabei werden manuelle Betriebsaufgaben reduziert und Ressourcen effizienter zugeteilt [14, 12]. Weitere Arbeiten adressieren prädiktive Skalierung, um Lastverläufe vorherzusagen und Ressourcen vorausschauend anzupassen [15]. Auch Reinforcement Learning wird zur dynamischen Steuerung der Lastverteilung und zur Erhöhung der Fehlertoleranz eingesetzt [16].
Ein weiterer Schwerpunkt liegt auf KI in DevOps- und CI/CD-Prozessen. Hierzu zählen die Vorhersage von Fehlern in Build- und Bereitstellungsprozessen sowie automatisierte Gegenmaßnahmen wie Rücknahme oder Anomaliealarme [8, 17, 18]. Ergänzend wird AIOps als Ansatz beschrieben, um komplexe Cloud-Infrastrukturen durch automatisierte Analyse und Monitoring besser beherrschbar zu machen [19, 13]. Sicherheitsbezogene Aspekte werden von Uddoh u. a. [20] ebenfalls aufgegriffen, etwa durch KI-basierte Erkennung von Bedrohungen und automatisierte Reaktionen im Plattformbetrieb. Das CNCF-Whitepaper verortet diese Richtung als „AI for Cloud Native Operations“ und beschreibt Assistenzwerkzeuge im operativen Cloud-Native Betrieb [1].
Kritisch ist festzuhalten, dass viele Arbeiten einzelne Anwendungsfälle isoliert betrachten oder stark werkzeug- bzw. monitoringgetrieben sind. Zudem existieren thematisch nahe Beiträge mit MLOps- oder Datenpipeline-Fokus, die den Plattformbetrieb aus Sicht von Platform Engineers nur indirekt adressieren und daher bewusst ausgeklammert wurden. Eine systematische, übergreifende Einordnung entlang typischer Aufgaben im Platform Engineering sowie eine Bewertung von Übertragbarkeit und praktischem Nutzen bleibt häufig offen. An dieser Stelle setzt die vorliegende Arbeit an.

3

Methodisches Vorgehen

Dieses Kapitel beschreibt das methodische Vorgehen dieser Arbeit. Es erläutert die zugrunde liegende Forschungslogik, die Auswahl des methodischen Ansatzes sowie die Verfahren zur Datenerhebung und -auswertung. Ziel ist es, ein wissenschaftlich fundiertes und zugleich praxisorientiertes Vorgehen aufzuzeigen, das eine systematische Untersuchung der Forschungsfragen ermöglicht. In Abschnitt 3.1 wird eine systematische Mapping Study (SMS) nach Petersen et al [21] angewendet. Abschnitt 3.2 definiert die spezifischen Ziele und konkreten Forschungsfragen dieser Arbeit, welche als Basis für die anschließende Analyse dienen. Darauf aufbauend beschreibt Abschnitt 3.3 den Prozess der Literaturanalyse. Dieser umfasst die Entwicklung einer Suchstrategie, die Definition von Ein- und Ausschlusskriterien, die Schneeballmethode sowie die Schritte der Datenextraktion und -synthese.

3.1. Vorgehen der Mapping Study

Für diese Arbeit wird eine Systematic Mapping Study (SMS) nach den Richtlinien von Petersen, Vakkalanka und Kuzniarz [21] durchgeführt. Diese Methodik dient dazu, den aktuellen Forschungsstand zu einem Themengebiet systematisch zu erfassen, zu kategorisieren und bestehende Forschungslücken zu identifizieren.
Das Vorgehen umfasst die Phasen Planung, Durchführung und Auswertung. In der Planungsphase werden die Forschungsfragen definiert und die Suchstrategie entwickelt, einschließlich der Auswahl relevanter wissenschaftlicher Datenbanken. Dabei wird gezielt nach bestimmten Keywords gesucht, um Publikationen zu finden, die KI-Anwendungen

im Kontext von Platform Engineering adressieren.
Um eine fundierte Datenerhebung und Auswertung sicherzustellen, werden einzelne Prinzipien einer Systematic Literature Review (SLR) nach Kitchenham und Charters [22] berücksichtigt. In der Durchführungsphase werden identifizierte Studien anhand festgelegter Ein- und Ausschlusskriterien geprüft. Zusätzlich wird das Schneeballverfahren nach Wohlin [23] eingesetzt, um die Literatursammlung zu erweitern. Alle Schritte werden dokumentiert, um die Nachvollziehbarkeit und Reproduzierbarkeit sicherzustellen. Die Auswertung erfolgt durch eine systematische Kategorisierung der Studien entlang zentraler Themenfelder des Platform Engineerings. Darauf aufbauend werden Muster, Trends und Forschungslücken identifiziert.
Die Ergebnisse der Mapping Study bilden die Grundlage für die anschließende Analyse der Bosch Digital Manufacturing Platform sowie für die Entwicklung eines Frameworks zur Bewertung von KI-Potenzialen in Cloud-Native Plattformumgebungen.

3.2. Forschungsfragen

Die Forschungsfragen werden durch ein strukturiertes methodisches Vorgehen beantwortet. Jede Forschungsphase ist darauf ausgelegt, das Verständnis über den Einsatz von Künstlicher Intelligenz im Platform Engineering schrittweise zu vertiefen. Die Mapping Study dient der Beantwortung der ersten beiden Forschungsfragen, indem sie eine systematische Übersicht über bestehende KI-Ansätze und deren Anwendungsfelder liefert. Auf Grundlage der Ergebnisse der Mapping Study zielt die dritte Forschungsfrage darauf ab, ein praxisorientiertes Framework zur Beantwortung und Übertragbarkeit von KI-Lösungen zu entwickeln. Die Arbeit ist entlang der folgenden Forschungsfragen strukturiert:

RQ1: Welche typischen Anwendungsfelder (Use Cases) und Herausforderungen bestehen im Platform Engineering, in denen KI-Technologien potenziell Mehrwert bieten können?
Ziel: Systematische Erfassung und Kategorisierung relevanter Use Cases.

RQ2: Welche KI-Technologien (einschließlich Frameworks und Tools) finden derzeit im Platform Engineering Anwendung, und welche Formen des maschinellen Lernens und Algorithmen kommen dabei zum Einsatz?
Ziel: Erstellung einer Übersicht über vorhandene KI-Ansätze und deren typische Einsatzkontexte.

RQ3: Wie lassen sich die identifizierten KI-Lösungen auf typische Anwendungsfälle in Cloud-Native-Platform-Umgebungen übertragen und hinsichtlich ihres Mehrwertes und ihrer Umsetzbarkeit bewerten?
Ziel: Entwicklung eines Bewertungsschemas, das die Passung zwischen KI-Lösungen und spezifischen Platform-Use-Cases beschreibt und die praktische Umsetzbarkeit aufzeigt.

3.3. Literaturanalyse-Prozess

Der folgende Abschnitt beschreibt den Ablauf der Literaturanalyse im Rahmen der durchgeführten Mapping Study. Ziel ist es, den methodischen Prozess transparent darzustellen. Dazu werden zunächst die Suchstrategie und die Auswahlkriterien erläutert, gefolgt von der Anwendung der Schneeballmethode. Danach wird die Datenextraktion sowie die Datensynthese beschrieben.

3.3.1. Suchstrategie

Zur Durchführung der Mapping Study wurde eine systematische Suchstrategie angewendet, um relevante wissenschaftliche Publikationen zu identifizieren. Die Literaturrecherche erfolgte in den Datenbanken Google Scholar, SpringerLink, ScienceDirect und IEEE Xplore, da diese eine breite Abdeckung im Bereich Software Engineering, Cloud Native Technologien und KI bieten. Ziel der Suche war, eine möglichst vollständige Übersicht aktueller Forschungsarbeiten zu KI-Anwendungen im Platform Engineering zu erhalten. Dafür wurden gezielt Suchbegriffe und Kombinationen von Suchstrings verwendet, die zentrale Themen der Arbeit abbilden. Die Suchbegriffe waren hierbei: ”Platform Engineering“, ”Cloud-Native“, ÄIOps“, Ärtificial Intelligence“, ”Machine Learning“, ”MLOps“, ”DevOps“ und ”Kubernetes Cluster“.
Zur Transparenz und Vollständigkeit sind die exakten Suchstrings sowie ihre logischen Verknüpfungen im Anhang dokumentiert.

3.3.2. Auswahlkriterien

Um relevante Studien und Publikationen zu identifizieren, wurde ein systematischer Auswahlprozess durchgeführt, der auf klar definierten Ein- und Ausschlusskriterien basiert. Eine Studie wurde in die Analyse aufgenommen, wenn sie alle Einschlusskriterien

erfüllt und zugleich keinem der Ausschlusskriterien unterlag. Die vollständige Übersicht der Kriterien ist in Tabelle 3.1 dargestellt.

Tabelle 3.1: Literatur-Auswahlkriterien

Kriterium	Beschreibung
EK1	Die Publikation befasst sich mit dem Einsatz von KI oder Machine Learning im Kontext von Platform Engineering, Cloud-Native-Technologien, DevOps oder AIOps.
EK2	Die Studie beschreibt konkrete KI-Methoden, Anwendungen, Architekturen oder Use Cases, die sich auf Plattformumgebungen beziehen.
EK3	Die Arbeit ist wissenschaftlich fundiert, z.B. als Konferenz-, Journal- oder White Paper, auch wenn kein Peer-Review-Verfahren vorliegt.
EK4	Veröffentlichungen sind in englischer oder deutscher Sprache verfasst und nach 2020 erschienen.
AK1	Arbeiten, die keinen direkten Bezug zu KI im Platform Engineering oder verwandten Domänen aufweisen.
AK2	Review-Paper oder systematische Übersichtsarbeiten, die keine eigenen empirischen oder technischen Beiträge enthalten.
AK3	Studien ohne nachvollziehbare methodische Grundlage oder ohne Beschreibung der verwendeten KI-Techniken.
AK4	Arbeiten mit primärem Fokus auf MLOps, insbesondere auf Infrastruktur, Deployment und Lebenszyklusmanagement von ML-Modellen für Data-Science-Workflows, ohne direkten Bezug zur operativen Unterstützung von Platform Engineers (AI for Ops).

3.3.3. Schneeballmethode

Zur Ergänzung der systematischen Suche wurde eine Schneeballmethode nach den Leitlinien von Wohlin [**wohlinGuidelinesSnowballingSystematic2014**] angewendet. Dabei erfolgte sowohl eine Rückwärtssuche als auch eine Vorwärtssuche. Als Ausgangspunkt dienten vier relevante Paper, auf deren Basis zwei Iterationen der Vorwärts- und Rückwärtssuche durchgeführt wurden. Die neu gefundenen Publikationen wurden nach denselben Ein- und Ausschlusskriterien geprüft. Der Prozess wurde beendet, sobald keine weiteren relevanten Studien identifiziert werden konnten.

3.3.4. Datenerhebung aus den Studien

Zur Sicherstellung von Konsistenz und Nachvollziehbarkeit wurde ein strukturierter Prozess zur Datenerhebung aus den Studien umgesetzt. Hierzu wurde eine eigene Extraktionsvorlage entwickelt, die die wesentlichen Merkmale der identifizierten Studien erfasst. Diese Merkmale wurden anschließend entlang von fünf zentralen Dimensionen kategorisiert, wie in Tabelle 3.2 dargestellt.

Tabelle 3.2: Kategorisierung der Datenerhebung

Dimension	Beschreibung
Forschungskontext	Beschreibt Ziel, Umfang und Art der Studie.
KI-Ansatz und Methode	Erfasst die verwendeten KI- oder ML-Verfahren.
Platform-Domänen	Ordnet den Beitrag einem Bereich des Platform Engineerings zu.
Ergebnisse und Use-Cases	Fasst die zentralen Erkenntnisse, Anwendungsfälle oder Evaluationsergebnisse zusammen.
Forschungslücke	Dokumentiert identifizierte Limitationen und Ansätze für zukünftige Arbeiten.

Die Extraktion erfolgte qualitativ, wobei relevante Textpassagen und zentrale Aussagen aus jeder Publikation manuell erfasst und den entsprechenden Dimensionen zugeordnet wurden

3.3.5. Datenanalyse und Kategorisierung

Nach der Datenerhebung wurden die Ergebnisse zusammengeführt und ausgewertet, um zentrale Themen, Muster und Forschungslücken zu erkennen. Die ausgewählten Studien wurden nach ihren Inhalten und Schwerpunkten strukturiert und den Forschungsfragen zugeordnet. Zur besseren Übersicht erfolgte die Kategorisierung der Arbeit entlang wichtiger Bereiche des Platform Engineerings. Innerhalb dieser Kategorien wurden die identifizierten KI-Ansätze, Anwendungsfälle und Herausforderungen miteinander verglichen, um wiederkehrende Trends sichtbar zu machen. Die Ergebnisse der Datenanalyse und Kategorisierung werden anschließend in Form einer Mapping Study dargestellt. Diese Übersicht zeigt, in welchen Themenfelder bereits Forschungsschwerpunkte existieren und wo noch Forschungslücken bestehen.

4

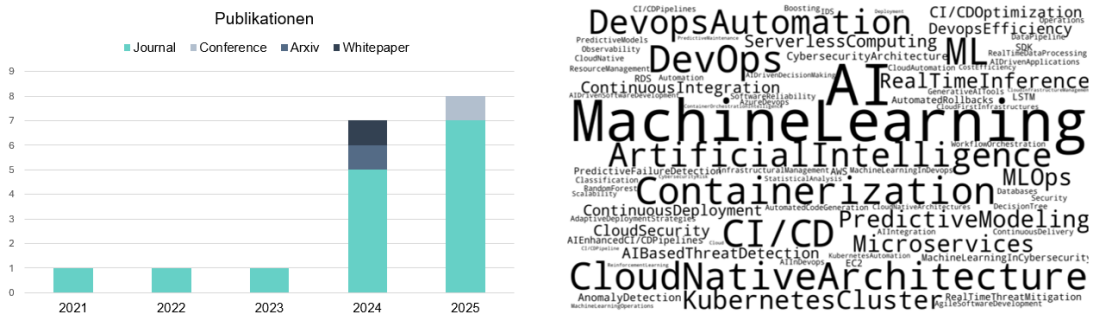
Ergebnisse der Literaturanalyse

Dieses Kapitel präsentiert die Ergebnisse der im methodischen Vorgehen beschriebenen Literaturuntersuchung. Zunächst werden in Abschnitt 4.1 die quantitativ erhobenen Merkmale der ausgewählten Publikationen ausgewertet. Abschnitt 4.2 stellt anschließend die Ergebnisse der eigentlichen Mapping Study vor, indem die Arbeiten systematisch klassifiziert und Muster sichtbar gemacht werden. Darauf aufbauend fasst Abschnitt 4.3 die identifizierten KI-Anwendungen im Platform-Engineering zusammen und leitet ein erstes strukturiertes Matching in Form eines konzeptionellen Frameworks ab. Die dargestellten Ergebnisse bilden die Grundlage für die Beantwortung der Forschungsfragen in Kapitel 6.

4.1. Quantitative Analyse

Zur Einordnung des untersuchten Forschungsfeldes wurden zunächst grundlegende Merkmale der insgesamt 18 Studien analysiert. Abbildung 4.1 a) zeigt die jährliche Verteilung der Publikationen sowie die Zuordnung zu verschiedenen Publikationstypen. Zwischen 2021 und 2023 erscheinen nur wenige Arbeiten (insgesamt drei), während ab 2024 ein deutlicher Anstieg sichtbar wird. Im Jahr 2024 wurden sieben und 2025 wurden acht Publikationen identifiziert, überwiegend Journalartikel, ergänzt durch jeweils ein Konferenzbeitrag, ArXiv-Paper und Whitepaper. Dies weist auf ein zunehmendes wissenschaftliches Interesse am Einsatz von KI in Cloud-Native- und Platform-Engineering-Kontexten hin. Das lässt sich auch unter anderem auf Grund des ChatGPT/GenAI Hype in den letzten beiden Jahren erklären [19].

Ergänzend dazu zeigt eine Word Cloud (Abbildung 4.1 b) die am häufigsten vorkommenden Keywords aus allen Publikationen. Die Visualisierung bietet einen schnellen Überblick über zentrale thematische Schwerpunkte der Literatur und unterstützt die anschließende inhaltliche Analyse.



a) Verteilung ausgewählter Publikationen nach Jahr und Typ

b) Ergebnisse der Themenmodellierung aus den Keywords der 21 ausgewählten Artikel

Abbildung 4.1: Jährliche und thematische Verteilung der DevOps AI-Forschung

Diese Betrachtung bildet die Grundlage für die folgenden Unterkapitel, in denen die Studien hinsichtlich ihrer inhaltlichen Merkmale detaillierter ausgewertet werden.

4.1.1. Anwendungsbereiche der KI im Platform Engineering

Im ersten Schritt der quantitativen Analyse wurden die insgesamt 18 identifizierten Studien hinsichtlich ihrer Hauptanwendungsbereiche untersucht. Dazu wurden die ausgewählten Bereiche in vier Kategorien unterteilt: Optimierung von CI/CD-Pipelines, Ressourcen- und Workload-Optimierung, Sicherheits- und Bedrohungserkennung sowie Betrieb und Orchestrierung der Plattform (GenOps).
Die Auswertung zeigt, dass Ressourcen- und Workload-Optimierung mit sieben Publikationen am häufigsten als dominanter Use Case auftritt. Ein plausibler Grund ist, dass Maßnahmen in diesem Bereich häufig einen direkten, messbaren Hebel auf Kosten und Performance haben (z.B. bessere Ressourcen-Auslastung oder stabilere Laufzeiten) [24]. An zweiter Stelle folgt der Bereich Betrieb und Orchestrierung mit fünf Publikationen. Dies deutet darauf hin, dass KI-Ansätze besonders im laufenden Betrieb relevant sind, etwa zur Unterstützung von Monitoring oder der automatisierten Steuerung von Plattformkomponenten. [18] Die Optimierung von CI/CD-Pipelines und Sicherheits- und Bedrohungserkennung sind mit jeweils drei Publikationen vertreten. Das lässt sich daraus erklären, dass diese beiden Anwendungsfelder zwar in vielen Publikationen thematisiert werden, jedoch seltener als primärer Fokus der Studie dienen. Die Optimierung der CI/CD-Pipeline wird oft als Teilaspekt in ein größeres Gesamtbild der Infrastruktur-Transformation behandelt [13].

Abbildung 4.2 zeigt diese Verteilung. Zu beachten ist, dass viele Studien mehr als einen Bereich ansprechen. Für die Vergleichbarkeit wurde jedoch jeweils der dominante Use Case pro Publikation ausgewählt. In der Praxis überschneiden sich die Anwendungsbereiche häufig, da KI-Lösungen oft mehrere Aufgaben gleichzeitig unterstützen.

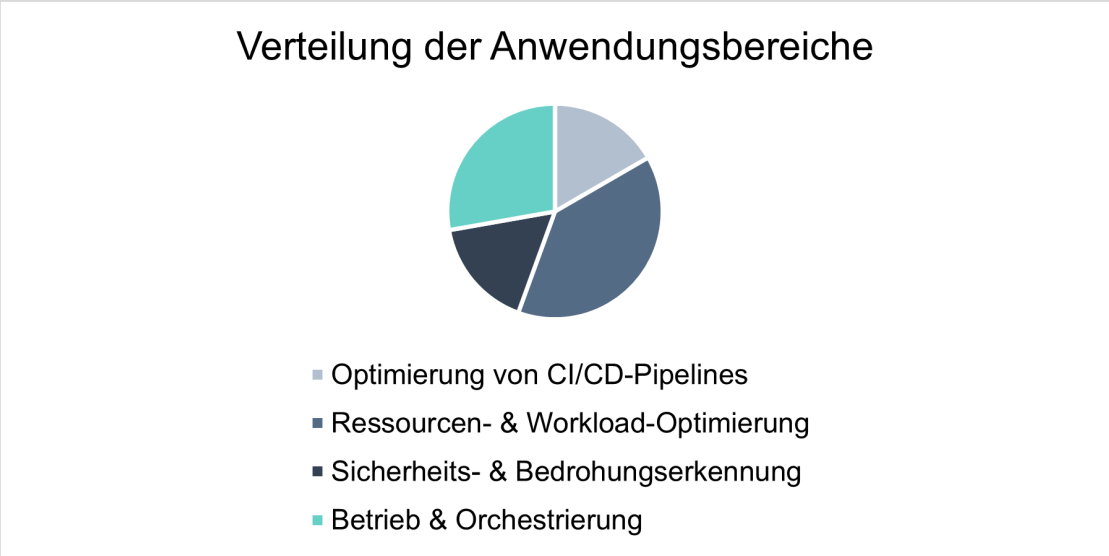


Abbildung 4.2: Verteilung der Anwendungsbereiche

4.1.2. Herausforderungen der KI-Integration im Platform Engineering

Im nächsten Schritt wurden die in den Publikationen beschriebenen Herausforderungen analysiert, die beim Einsatz von KI im Platform Engineering auftreten. Die analysierten Paper wurden fünf Kategorien zugeordnet. Die prozentuale ist in Abbildung 4.3 dargestellt.
Die Auswertung zeigt, dass Ressourcenverbrauch und Kosten mit 94 % (17/18) am häufigsten als Herausforderung genannt werden. Ebenfalls häufig werden Skalierbarkeit, Latenz und Monitoring mit 83 % (15/18) sowie KI-Governance, Datenschutz und Compliance mit 83 % (15/18) adressiert. Integrationskomplexität und Abhängigkeiten werden in 78 % (15/18) der Studien thematisiert. Die fünfte Kategorie, Datenqualität, Datenverfügbarkeit und Heterogenität, wird in 72 % (13/18) der Arbeiten als Herausforderung genannt.
Diese Ergebnisse deuten auf ein Spannungsfeld hin: KI wird zwar eingesetzt, um Plattformen effizienter und kostengünstiger zu betreiben, erzeugt jedoch durch Training, Inferenz und zusätzliche Observability- und Datenpipelines oft selbst signifikante Ressourcen- und Betriebskosten [1]. Die hohe Nennung von KI-Governance, Datenschutz und Compliance legt zudem nahe, dass der produktive KI-Einsatz in Plattformen häufig weniger

an der reinen technischen Machbarkeit scheitert, sondern stark durch Anforderungen an Datenzugriff, Nachvollziehbarkeit und Regelkonformität begrenzt wird [17].

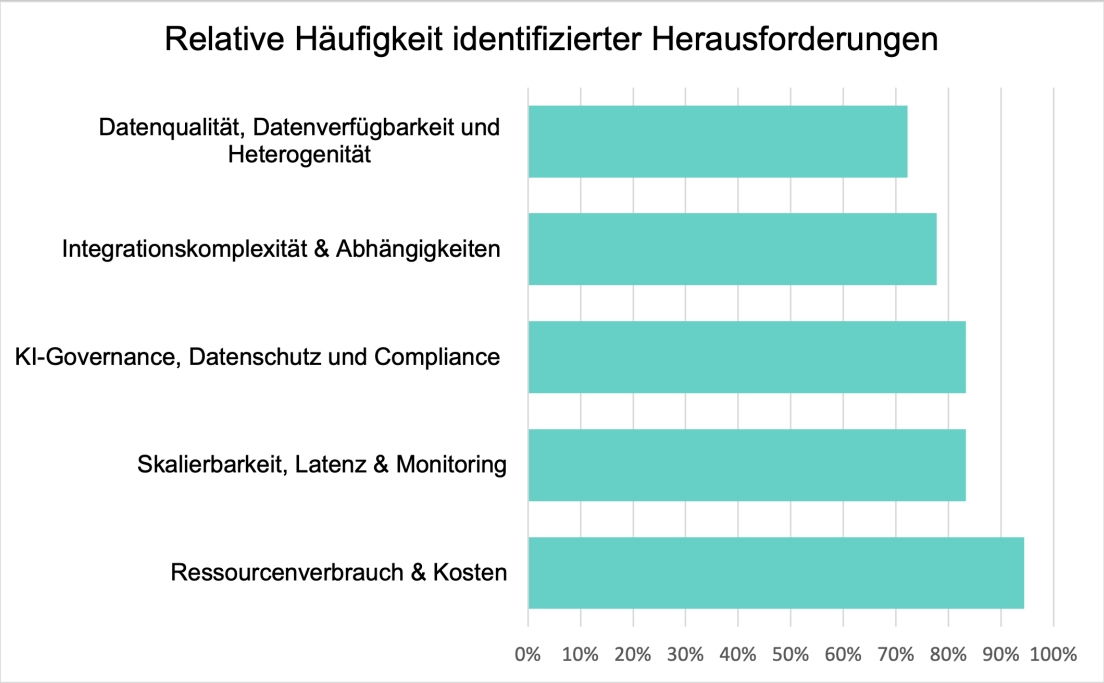


Abbildung 4.3: Relative Häufigkeit der Herausforderungen

4.1.3. Formen des maschinellen Lernens

Ein weiterer Bestandteil der quantitativen Analyse umfasst die in den Publikationen verwendeten Formen bzw. Lernparadigmen des maschinellen Lernens. Dabei wurde unterschieden zwischen (1) explizit benannt, (2) implizit anhand der beschriebenen Methode ableitbar und (3) nur kurz erwähnt ohne weitere methodische Ausführung. Insgesamt wurden die drei grundlegenden Paradigmen identifiziert: Supervised Learning, Unsupervised Learning und Reinforcement Learning.
Die Auswertung zeigt, dass Supervised Learning in den meisten Arbeiten eine zentrale Rolle spielt. Es wird in vier Publikationen ausdrücklich beschrieben, elfmal implizit erkennbar und in zwei kurz erwähnt. Reinforcement Learning wird insgesamt siebenmal explizit genannt und in vier weiteren Arbeiten erwähnt. Unsupervised Learning tritt mit drei expliziten und sieben impliziten Nennungen seltener auf, ist jedoch ebenfalls präsent.
Die Verteilung ist in der folgenden Abbildung 4.4 dargestellt.

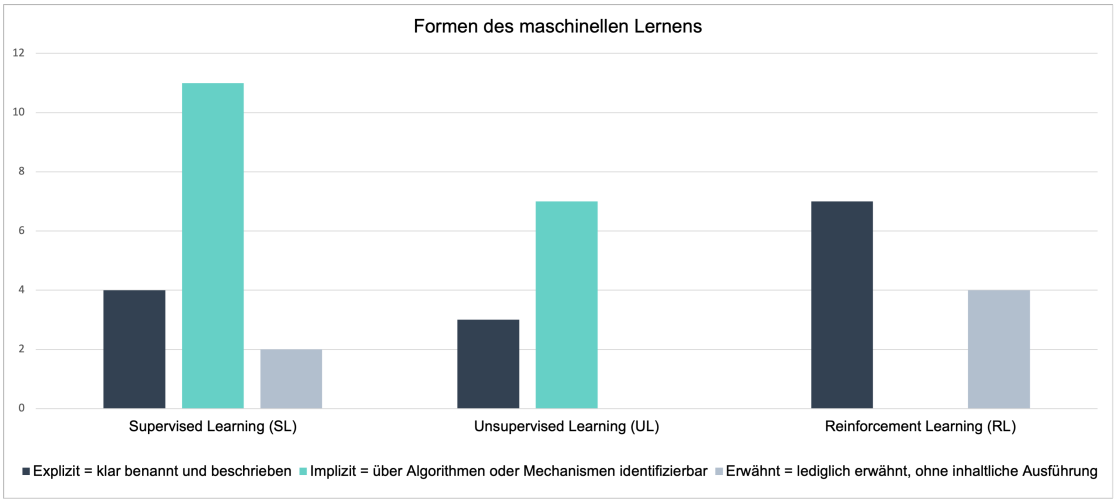


Abbildung 4.4: Formen des maschinellen Lernens

Auffällig ist die deutliche Differenz zwischen explizit und implizit erkennbaren Anwendungen, insbesondere bei Supervised und Unsupervised Learning. Dies zeigt, dass viele Publikationen die entsprechenden Paradigmen beschreiben, ohne die zugrunde liegende Lernform ausdrücklich zu benennen. Ein möglicher Erklärungsansatz für die Dominanz von Supervised Learning ist, dass in Plattformumgebungen häufig gut messbare Zielgrößen und bereits beschriftete Verlaufsdaten vorliegen (z.B. Störungsmeldungen, Support-Tickets oder Metriken mit bekanntem Ergebnis), wodurch sich SL-Ansätze besonders gut anwenden und bewerten lassen [17].

4.1.4. Verwendete Algorithmen

Die in den Publikationen verwendeten Algorithmen lassen sich den jeweiligen Lernparadigmen zuordnen. Die Analyse zeigt, dass eine Vielfalt an KI- und ML-Verfahren im Platform-Engineering-Kontext eingesetzt wird. Für eine systematische Bewertung wurde eine eigene Kategorisierung entwickelt. Den Ausgangspunkt bildet die Tabelle aus Enemosah [8], in der verschiedene KI-Techniken für Testfallpriorisierung beschrieben werden. Die Einteilung wurde für den breiteren Kontext dieser Arbeit angepasst.
Auf dieser Basis wurden vier Kategorien definiert, die in Kapitel 2.1 kurz beschrieben sind. Anschließend folgt die quantitative Auswertung der identifizierten Methoden und Algorithmen. Dies zeigt, wie häufig die jeweiligen Verfahren in den Publikationen genannt werden.
Die Ergebnisse zeigen ein deutliches Übergewicht von Deep Learning / neuronalen Netzen, die in 89% (16/18) der Publikationen eingesetzt oder thematisiert werden. Klassische Klassifikation/Regression tritt mit 44% (8/18) ebenfalls häufig auf. Ensemble-

und baumbasierte Verfahren werden in 39% (7/18) der Arbeiten genannt. Clustering ist mit 33% (6/18) vertreten. Insgesamt zeigt sich, dass insbesondere neuronale Netze die dominierende Methodenklasse bilden. Die Dominanz von Deep-Learning-Verfahren kann darauf hindeuten, dass in der Forschung häufig komplexere Modelle bevorzugt werden, weil sie mit heterogenen Daten (z.B. Logs, Metriken, Text) flexibel umgehen können. Gleichzeitig deutet sie auch auf ein Spannungsfeld hin: Für einige Aufgaben könnten einfachere und besser erklärbare Modelle bereits ausreichen, werden in der Literatur aber seltener als Schwerpunkt gesetzt [8, 17].
Diese Verteilung ist in der Abbildung 4.5 dargestellt.

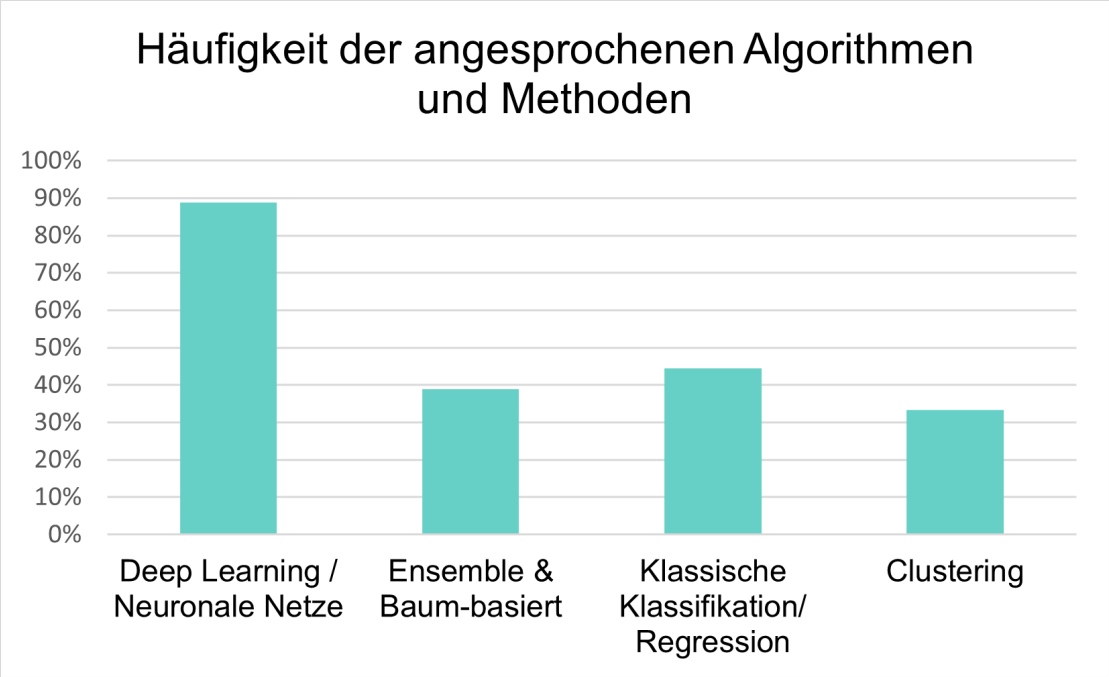


Abbildung 4.5: Verteilung der Algorithmen und angesprochenen Methoden

4.2. Ergebnisse der Mapping Study

Um die Beziehungen innerhalb der KI-Forschung im Platform Engineering weiter zu verdeutlichen, baut dieser Abschnitt auf den vorherigen Erkenntnissen auf und präsentiert detaillierte Kreuztabellenanalysen. Diese Zuordnungen untersuchen Beziehungen zwischen Anwendungsfeldern und Herausforderungen (Abschnitt 4.2.1), Lernparadigmen und Algorithmen (Abschnitt 4.2.2) sowie Anwendungsfelder und Datenquellen (Abschnitt 4.2.3). Die Ergebnisse werden als Bubble-Chart-Diagramme visualisiert, in denen die Blasengröße die Häufigkeit der jeweiligen Zuordnung (n) widerspiegelt. Die Häufigkeiten (n) geben an, in wie vielen der betrachteten Studien die jeweilige Zuordnung vorkommt. Pro Studie wird eine Zuordnung je Kombination höchstens einmal gezählt, unabhängig davon, wie häufig sie im Text erwähnt wird. Ziel ist es, durch diese systematischen Mappings tiefere Einblicke in Struktur und Schwerpunkte der aktuellen Forschung zu gewinnen.

4.2.1. Zusammenspiel der Anwendungsfelder und Herausforderungen

Die Abbildung 4.6 zeigt das Zusammenspiel zwischen den identifizierten Use Cases und den zentralen Herausforderungen im Platform Engineering. Im Gegensatz zur Abbildung 4.2 wurden hier alle in den analysierten Studien genannten Anwendungsbereiche berücksichtigt und den jeweils adressierten Herausforderungen zugeordnet. Die Häufigkeiten geben an, wie oft eine bestimmte Kombination in der Literatur thematisiert wurde.

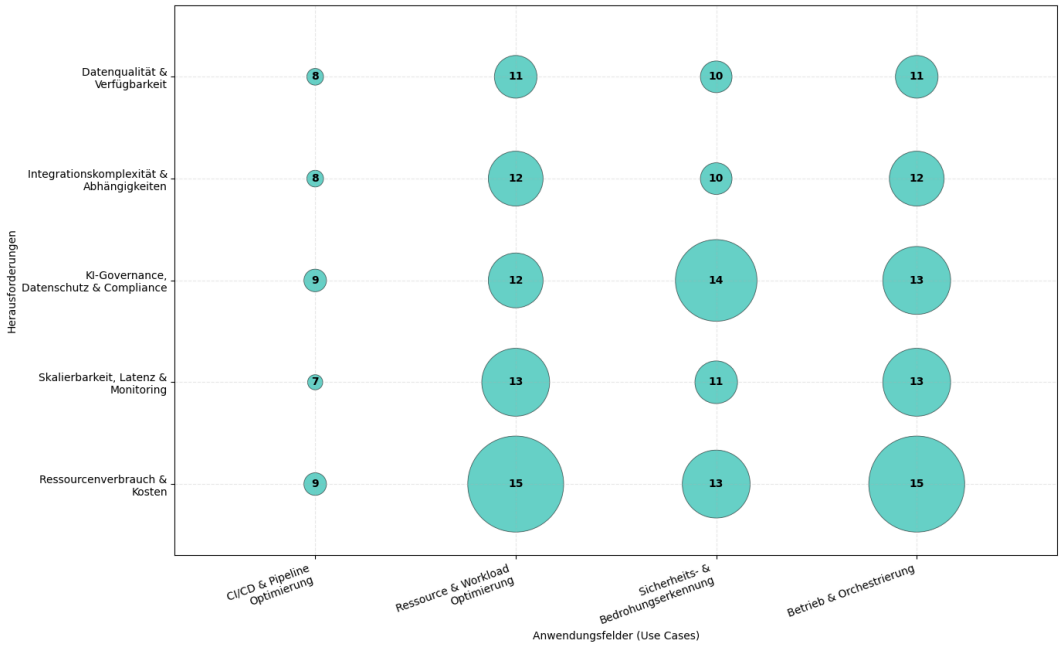


Abbildung 4.6: Korrelation zwischen Anwendungsfelder und Herausforderungen

Besonders deutlich wird die starke Verknüpfung der Use Cases Ressource- und Workload-Optimierung sowie Betrieb und Orchestrierung mit nahezu allen Herausforderungskategorien. Die Ressourcen- und Workload-Optimierung weist über alle Bereiche hinweg hohe Werte auf (n=15, 13, 12, 12, 11). Diese breite Korrelation ist plausibel, da Ressourcen- und Workload-Optimierung einen besonders direkten Bezug zu Kosten hat und sich Effekte häufig über die Cloud-Abrechnung unmittelbar sichtbar machen. KI-gesteuertes Auto-Scaling vermeidet die Überbereitstellung von Ressourcen und senkt dadurch die monatlichen Abrechnungskosten direkt [15]. Gleichzeitig wird deutlich, dass Echtzeit-Inferenz zwar Genauigkeit und Latenz verbessert, jedoch durch höheren CPU-/Memory-Verbrauch die Ressourcen- und Kostenbelastung erhöht und damit eine Abwägung zwischen Modellqualität und Betriebsaufwand erforderlich macht [25].

Ein sehr ähnliches Muster zeigt sich im Bereich Betrieb & Orchestrierung, der ebenfalls in allen Herausforderungskategorien hohe Häufigkeiten erreicht (n=15, 13, 13, 12, 11). Das deutet darauf hin, dass KI im Plattformbetrieb häufig nicht als Einzellösung betrachtet wird. Stattdessen ist sie meist in mehrere Betriebsbausteine eingebettet, z.B. Monitoring, Deployment, Orchestrierung und Governance. Damit werden in diesem Bereich oft mehrere Herausforderungen gleichzeitig berührt, etwa Kosten, Skalierung, Integration und datenbezogene Voraussetzungen.

Der Use Case Sicherheits- und Bedrohungserkennung zeigt seine höchste Ausprägung im Bereich KI-Governance, Datenschutz und Compliance (n=14) sowie bei Ressour-

cenverbrauch und Kosten (n=13). Auch die übrigen Herausforderungen weisen weiterhin hohe Werte auf (n=11, 10, 10). Dies deutet darauf hin, dass sicherheitsbezogene KI-Ansätze neben Governance-Themen häufig auch Monitoring, Integrationsprozesse und die zugrunde liegende Datenlage betreffen.
Die CI/CD- & Pipeline-Optimierung zeigt insgesamt die niedrigsten Werte, bleibt aber über alle Herausforderungen hinweg relativ konstant (n=9, 7, 9, 8, 8). KI wird hier vor allem eingesetzt, um einzelne Schritte wie Builds, Tests oder Deployments zu verbessern. Im Vergleich zu ressourcen- oder betriebsnahen Use Cases werden jedoch weniger Herausforderungen gleichzeitig berührt. Kosten, Tool-Abhängigkeiten und Governance-Fragen bleiben trotzdem relevant, etwa durch zusätzlichen Rechenaufwand und die notwendige Nachvollziehbarkeit automatisierter Entscheidungen [17].
Insgesamt verdeutlicht die Verteilung der Häufigkeiten, dass KI-Anwendungen im Platform Engineering besonders dort adressiert werden, wo betriebliche Effizienz und Automatisierung direkt mit Kosten, Skalierung, Governance und Integrationsfragen zusammenwirken. Die Häufungen zeigen somit, welche Themen in der Forschung besonders im Fokus stehen und in welchen Bereichen KI-Lösungen aktuell besonders häufig diskutiert werden.

4.2.2. Zusammenspiel der Lernparadigmen und Algorithmen

Die Abbildung 4.7 visualisiert die Korrelation zwischen den in den analysierten Studien verwendeten Lernparadigmen des maschinellen Lernens und den eingesetzten Algorithmen. Einige Kombinationen werden in der Darstellung nicht ausgewiesen (n=0 bzw. ohne Blase), da sie in den betrachteten Studien nicht eindeutig als eigenständige Zuordnung berichtet wurden.

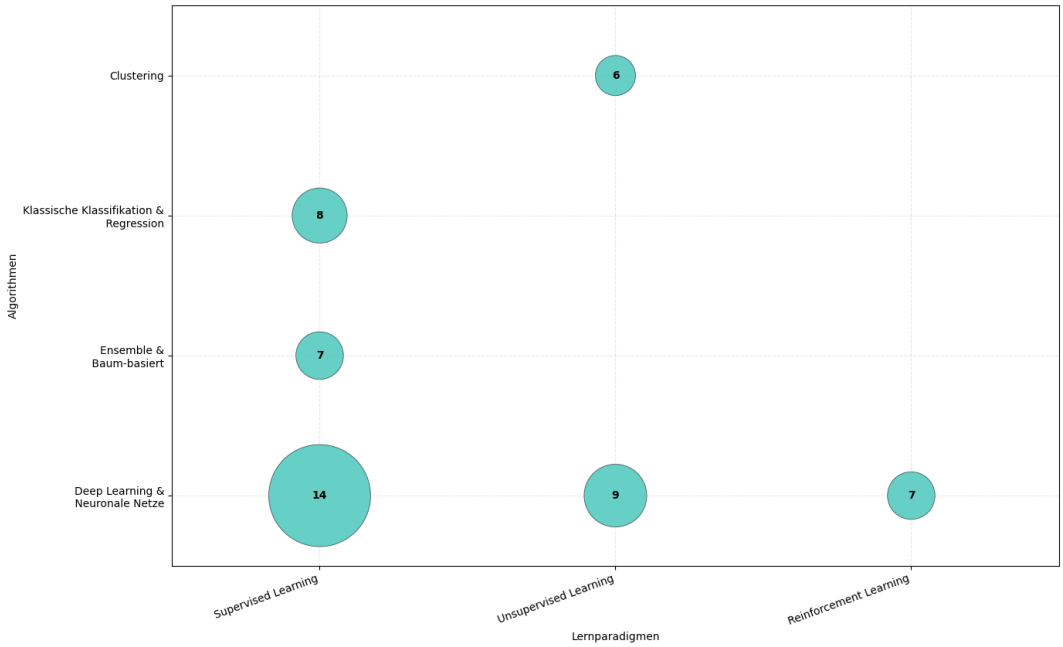


Abbildung 4.7: Korrelation zwischen Lernparadigmen und Algorithmen

Supervised Learning weist die größte Bandbreite an möglichen Algorithmen auf. Dies liegt daran, dass überwachte Lernverfahren sowohl tiefen neuronalen Netzen als auch klassischen Klassifikations- und Regressionsmodellen sowie ensemblebasierten Ansätzen zugrunde liegen. Der hohe Anteil an SL-Kombinationen (n=14 DL/NN, n=8 Klassifikation/Regression, n=7 Ensemble/baum-basiert) zeigt, dass SL sehr flexibel einsetbar ist. Solange gelabelte Daten vorliegen, können verschiedene Algorithmen angewendet werden. Ein wesentlicher Grund für die Dominanz neuronaler Netze ist ihre Fähigkeit, komplexe und nicht-lineare Beziehungen in großen Datenmengen zu erfassen, die für einfachere Modelle nicht erkennbar sind [17].
Unsupervised Learning zeigt hingegen ein engeres Spektrum. Die Ergebnisse verdeutlichen, dass UL hauptsächlich in Kombination mit Deep Learning eingesetzt wird (n=9) oder mit Clustering-Methoden (n=6). Dies ist erwartungsgemäß, da UL in den untersuchten Studien vor allem zur Mustererkennung und Anomalieanalyse eingesetzt wird und Clustering hierbei eine zentrale Rolle einnimmt, wie in [8] beschrieben. Der praktische Einsatz ist jedoch erschwert, da sich das als „normal“ erlernte Systemverhalten in dynamischen CI/CD-Umgebungen durch neue Funktionen oder Architekturänderungen häufig verschiebt und dadurch vermehrt Fehlalarme ausgelöst werden können, was den operativen Nutzen solcher Ansätze einschränkt [17].
Reinforcement Learning tritt zwar in mehreren Studien auf, wird jedoch selten algorithmisch konkretisiert. RL wird vor allem für dynamische Optimierungsaufgaben eingesetzt. Durch das Lernen auf Basis beobachteter Systemzustände und Feedback

können RL-Modelle geeignete Aktionen auswählen. Dadurch können sie zur Stabilisierung des Betriebs und zur Verbesserung der Systemleistung beitragen. Einzelne Arbeiten nutzen hierfür Deep-RL-Ansätze wie Deep Q-Networks [16], insbesondere wenn komplexe Zustandsräume berücksichtigt werden müssen. Ein zentrales Problem von RL ist jedoch die hohe Komplexität bei der Modellierung interagierender Systemkomponenten sowie der damit verbundene Rechenaufwand, wodurch eine Überführung in den produktiven Betrieb häufig erschwert wird [16]. Dies verdeutlicht, warum Reinforcement-Learning-Verfahren in der Praxis bislang nur begrenzt eingesetzt werden. Insgesamt bleibt die Zuordnung zu spezifischen Algorithmen jedoch begrenzt, weshalb viele Kombinationen in der Tabelle nicht belegt sind.

4.2.3. Zusammenspiel der Anwendungsfelder und Datenquellen

In diesem Unterkapitel wird analysiert, welche Datenquellen in den identifizierten Anwendungsfeldern (Use Cases) genutzt werden. Dafür wurden alle angesprochenen Anwendungsfelder pro Studie mit den jeweils verwendeten Datenquellen verknüpft. Abbildung 4.8 zeigt die resultierenden Häufigkeiten (n) je Kombination.

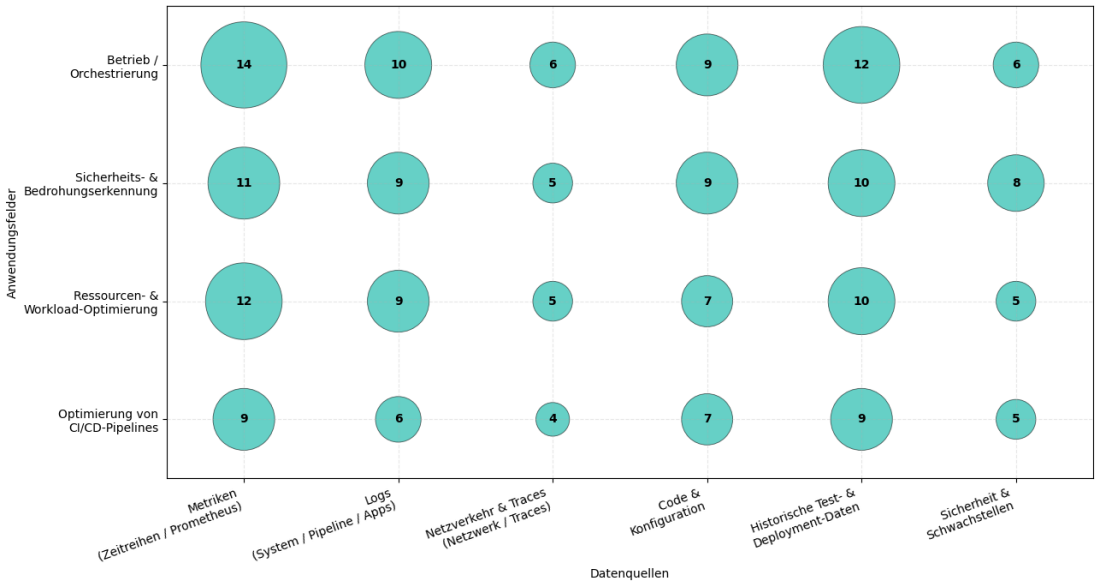


Abbildung 4.8: Korrelation zwischen Anwendungsfeldern und Datenquellen

Metriken (z. B. Zeitreihen aus Prometheus oder CloudWatch) beschreiben den quantitativen Systemzustand wie CPU, Speicher oder Latenzen. Entsprechend treten sie in nahezu allen Use Cases stark auf, besonders in Betrieb/Orchestrierung (n=14) sowie Ressourcen- und Workload-Optimierung (n=12), da diese Aufgaben auf kontinuierli-

chem Monitoring und wiederkehrenden Lastmustern basieren [12, 15].
Logs (System-, Pipeline- und Applikationslogs) sind ereignisbasierte, häufig unstrukturierte Daten und werden vor allem für Diagnose, Fehlerklassifikation und Ursachenanalyse genutzt. Im Mapping zeigen sie über nahezu alle Use Cases hinweg eine zentrale Rolle (typisch n=6-10), mit hohen Werten in Betrieb/Orchestrierung (n=10) und sicherheitsnahen Szenarien (n=9), da Betriebsstörungen, Fehlkonfigurationen oder Angriffsindikatoren meist zuerst in Logdaten sichtbar werden [26].
Netzwerkdaten und Traces ergänzen diese Sicht um Kommunikationsbeziehungen und Laufzeitpfade verteilter Systeme (z. B. Flow-Daten oder OpenTelemetry-Traces). Die Ausprägung ist insgesamt niedriger als bei Metriken und Logs, aber konsistent in Betrieb/Orchestrierung (n=6) sowie Sicherheits- und Bedrohungserkennung (n=5) vorhanden. Das deutet darauf hin, dass diese Daten vor allem dann relevant werden, wenn Ursachen nicht lokal erklärbar sind (z. B. in Microservice-Architekturen) oder wenn Anomalien über Kommunikationsmuster erkannt werden sollen [1].
Code und Konfiguration (Quellcode, Container-Artefakte, IaC-Definitionen) werden insbesondere dann wichtig, wenn KI nicht nur Symptome, sondern Änderungen und Konfigurationsursachen bewerten soll. Im Mapping zeigt sich eine breite Nutzung über alle Use Cases (n=7-9). Auffällig ist die hohe Ausprägung bei Betrieb/Orchestrierung und Security (je n=9), was zu typischen Plattformproblemen wie riskanten Änderungen, unsicheren Defaults oder Fehlkonfigurationen passt [13].
Historische Test- und Deployment-Daten (Build-Historien, Testergebnisse, Deployment-Verläufe) sind erwartungsgemäß stark vertreten, da viele Ansätze aus wiederkehrenden Mustern der Vergangenheit lernen. Besonders Betrieb/Orchestrierung (n=12) sowie CI/CD-Optimierung (n=9) und Ressourcenoptimierung (n=10) zeigen hohe Werte. Das ist plausibel, weil Stabilitäts- und Effizienzentscheidungen in der Praxis häufig auf Release-Historien, Fehlerraten und Zeitverläufen aufbauen [17].
Sicherheits- und Schwachstellendaten (z. B. Vulnerability-Scans, Threat-Feeds, Policy-Checks, Verhaltensmuster) konzentrieren sich erwartungsgemäß auf Sicherheits- und Bedrohungserkennung (n=8). Gleichzeitig sind sie auch im Betrieb/Orchestrierung sichtbar (n=6), was darauf hinweist, dass Security in Plattformumgebungen zunehmend als Betriebsaufgabe mit Observability- und Lebenszyklusdaten zusammenwächst [27].
Deutlich wird, dass der Use case Betrieb und Orchestrierung die stärksten Überschneidungen mit nahezu allen Datenquellen hat, weil hier Monitoring-Daten und Änderungsdaten zusammelaufen. Für die KI heißt das konkret, ohne diese Kombination entstehen eher Fehleralarme (Incidents), und die eigentliche Ursache lässt sich schlechter finden.

4.3. Matching-Framework

Aufbauend auf den quantitativen Ergebnissen aus Abschnitt 4.1 sowie den Mustern der Mapping Study in Abschnitt 4.2 wird in diesem Abschnitt ein konzeptionelles Matching-Framework abgeleitet. Ziel ist es, die identifizierten KI-Ansätze in übertragbare Anwendungsmuster für generische Use Cases im Platform Engineering zu strukturieren und damit eine nachvollziehbare Grundlage zu schaffen, um KI-Lösungen später hinsichtlich operativem Nutzen und Umsetzbarkeit einzuordnen.

4.3.1. Proaktives Ressourcen-Management

Das Muster Proaktives Ressourcen-Management beschreibt KI-gestützte Verfahren, um Infrastrukturressourcen vorausschauend und automatisiert an die erwartete Last anzupassen. Tabelle 4.1 fasst dieses Muster als konkrete Plattformaufgabe zusammen, bei der Instanzen oder Kubernetes-Pods (z. B. per Autoscaling/HPA) nicht nur reaktiv gesteuert werden. Auf Basis historischer Betriebsdaten wie CPU- und Speicherauslastung werden wiederkehrende Lastmuster erkannt und die Anzahl von Pods oder Instanzen frühzeitig angepasst, bevor Engpässe auftreten.

Tabelle 4.1: Übersicht zum Muster Proaktives Ressourcen-Management

Proaktives Ressourcen-Management	**Beschreibung**
Konkreter Plattform-Task	Automatisierte Anpassung von Instanztypen (EC2/RDS) oder Kubernetes-Pods (HPA) basierend auf der Vorhersage von CPU-/RAM-Spitzen.
Empfohlene KI-Methode	Deep Learning (z..B. LSTM) zur Zeitreihenprognose oder Reinforcement Learning zur dynamischen Lastverteilung in Echtzeit.
KPI/Mehrwert	Ressourcenauslastung bis zu +25 % (RL vs. Heuristik) [15], Kosteneffizienz durch dynamische Allokation mit bis zu 55 % geringeren Cloud-Kosten [24].
Limitationen	Hoher Rechenaufwand für das Modelltraining (z..B. LSTMs), Cold-Start-Latenzen in Serverless-Umgebungen, benötigt ca. 2 Monate historische Daten [28, 15].
Einsatzvoraussetzungen	Mind. 2 Monate historische Zeitreihen-Telemetrie (feingranular, z..B. 1-15 Minuten) aus CloudWatch/Prometheus sowie automatisierter API-Schreibzugriff für Instanz- bzw. Skalierungsanpassungen [15].
Konkrete Tools	AWS CloudWatch, Boto3 SDK, Kubernetes HPA, KEDA (Event-Driven Autoscaling), Knative, Megalix [29, 1, 15, 13].

In der Literatur werden dafür vor allem Verfahren zur Auswertung zeitlicher Verläufe sowie lernbasierte Ansätze zur Entscheidungsunterstützung beschrieben [15]. Diese unterstützen die Ableitung von Skalierungsentscheidungen auf Basis historischer Auslastungsdaten. Der Nutzen im Betrieb zeigt sich in einer gleichmäßigeren Ressourcennutzung und in geringeren Kosten, da Ressourcen bedarfsgerechter bereitgestellt werden [24]. Die Umsetzung setzt jedoch voraus, dass ausreichend Verlaufsdaten aus dem Betrieb vorliegen und Eingriffe in die Skalierung automatisiert möglich sind. Zusätzlich steigt der Aufwand, wenn Modelle regelmäßig angepasst werden müssen oder Verzögerungen beim Start neuer Komponenten auftreten, etwa in serverlosen Umgebungen.

4.3.2. Automatisierte Release-Absicherung

Das Muster Automatisierte Release-Absicherung adressiert die Absicherung von Ausrollungen, indem fehlerhafte Versionen möglichst früh erkannt und automatisiert begrenzt werden. Die Tabelle 4.2 fasst dieses Muster als Plattformaufgabe zusammen. Laufzeitdaten wie Latenz und Fehlerraten werden während der Ausrollung kontinuierlich ausgewertet und bei auffälligen Abweichungen wird automatisch gestoppt oder auf eine stabile Vorgängerversion zurückgesetzt.

Tabelle 4.2: Übersicht zum Muster Automatisierte Release-Absicherung

Automatisierte Release-Absicherung	**Beschreibung**
Konkreter Plattform-Task	Automatisiertes Rollback bei Canary-Deployments (schrittweise Ausrollung) durch Überwachung von Abweichungen in Latenz und Fehlerraten (Release-Gating).
Empfohlene KI-Methode	Unsupervised Learning (Autoencoder) zur Anomalieerkennung oder Reinforcement Learning (z.B. DQN) zur Optimierung des Rollback-Zeitpunkts [17].
KPI/Mehrwert	Mean Time To Recovery (MTTR) bis zu -40 %; Systemverfügbarkeit bis zu +18 %; Vorfallerkennung um bis zu 35 % beschleunigt [12, 16, 17].
Limitationen	Risiko von False Positives (unnötige Rollbacks); zusätzlicher Validierungs- und Governance-Aufwand bei RL-Entscheidungen (z.B. Nachvollziehbarkeit/Erklärbarkeit).
Einsatzvoraussetzungen	Etablierte Baseline für Normalverhalten in Service-Mesh-gestützten Telemetriedaten (Traffic-Steuerung & Observability; Latenz, Fehlerraten) sowie automatisierte Rollback-Mechanismen und Echtzeit-Streaming der Deployment-Metriken.
Konkrete Tools	Spinnaker / Spinnaker AI, Harness, StackStorm, Dynatrace, LaunchDarkly, Sentry, Istio, Linkerd [29, 30, 13].

Verfahren zur Erkennung von Abweichungen vom normalen Systemverhalten kommen zum Einsatz, um problematische Änderungen frühzeitig zu identifizieren [17]. Als operativer Mehrwert wird in den betrachteten Studien eine verkürzte Wiederherstellungszeit sowie eine höhere Systemverfügbarkeit beschrieben, da fehlerhafte Versionen schneller erkannt und automatisiert zurückgenommen werden können [12, 16, 17]. In der Praxis zeigen sich jedoch Grenzen durch Fehlalarme, die unnötige Rücksetzungen auslösen können, sowie durch zusätzlichen Validierungs- und Abstimmungsaufwand. Für eine robuste Umsetzung sind daher eine stabile Referenz für das Normalverhalten sowie automatisierte Rücksetz-Mechanismen und eine kontinuierliche Erfassung relevanter Ausrollungsdaten erforderlich.

4.3.3. Intelligente Build-Fehlerdiagnose

Das Muster Intelligente Build-Fehlerdiagnose adressiert die Frage, wie Fehlschläge in CI-Pipelines frühzeitig erkannt und die Ursachenanalyse im Entwicklungsprozess unterstützt werden kann. Tabelle 4.3 fasst dieses Muster als Plattformaufgabe zusammen, bei der Build- und Testprotokolle aus der CI automatisiert ausgewertet werden, um wiederkehrende Fehlermuster zu erkennen und einzugrenzen. Durch die Auswertung historischer Build-Protokolle, Testergebnisse und Code-Merkmale werden Muster identifiziert, die auf bevorstehende Fehlschläge hinweisen, sodass Entwickler frühzeitig auf potenzielle Build-Probleme aufmerksam gemacht werden.

Tabelle 4.3: Übersicht zum Muster Intelligente Build-Fehlerdiagnose

Intelligente Build-Fehlerdiagnose	**Beschreibung**
Konkreter Plattform-Task	Früherkennung von Build-Fehlschlägen durch Analyse von CI-Logs (z.B. Jenkins/Git) sowie automatisierte Clusterung/Filterung von Log-Signaturen zur Ursachenanalyse.
Empfohlene KI-Methode	Supervised Learning (z.B. Random Forest, XGBoost) zur Klassifikation von Fehlerrisiken und Fehlertypen auf Basis historischer Build-Metadaten und Log-Features [12, 17].
KPI/Mehrwert	Deployment-Zeit bis zu -30 %; Vorhersagegenauigkeit ca. 87 %; Reduzierung der Fehlersuchzeit bis zu 40 % [12, 17].
Limitationen	Class Imbalance (wenige Fehlersamples); Konzeptdrift bei Änderungen an Build-Pipelines/Dependencies erfordert kontinuierliches Monitoring und Retraining.
Einsatzvoraussetzungen	Zentralisierte und konsistent strukturierte Build- und Test-Logs sowie eine ausreichend große Historie gelabelter Daten (Erfolg vs. Fehlertyp) inklusive stabiler Referenz auf Commit, Pipeline-Stage und Artefaktversion.
Konkrete Tools	K8sGPT, Jenkins X (mit AI-Plugins), CircleCI Insights, DeepCode, SonarQube, GitLab CI, GitHub Copilot [12, 8, 1, 30, 13].

Hier werden vor allem SL-Verfahren wie Random Forest und XGBoost eingesetzt. Sie werten frühere Build-Daten und Build-Logs aus und schätzen damit sowohl das Risiko eines Build-Fehlers als auch häufig die Art des Fehlers ab [12, 27, 17]. Der Nutzen im Betrieb liegt vor allem in schnelleren Rückmeldungen. Fehleranfällige Änderungen fallen früher auf, und die Fehlersuche wird erleichtert, da relevante Logstellen automatisch vorausgewählt werden. In den Studien zeigt sich dies unter anderem in kürzeren Bereitstellungszeiten und einer deutlich geringeren Zeit für die Fehlersuche [12, 17]. Eine Herausforderung ist die ungleiche Verteilung der Daten. Build-Fehler treten deutlich seltener auf als erfolgreiche Builds, wodurch die Verfahren fehlerhafte Builds schlechter erkennen können. Zusätzlich verändern sich typische Muster, wenn Pipeline-Schritte, Abhängigkeiten oder Build-Umgebungen angepasst werden. In solchen Fällen müssen die Verfahren regelmäßig überprüft und erneut angepasst werden. Für einen stabilen Einsatz sind daher zentral gesammelte und einheitlich strukturierte Build- und Test-Logs notwendig. Zudem wird eine ausreichende und große Historie benötigt, die Builds eindeutig mit Commit, Pipeline-Schritt und Artefaktversion verknüpft.

4.3.4. Risikobasiertes Schwachstellen- und Compliance-Management

Das Muster Risikobasiertes Schwachstellen- und Compliance-Management wird in der Literatur häufig unter den Begriffen DevSecOps oder SecurityOps eingeordnet. Es beschreibt KI-gestützte Ansätze, um Sicherheitsereignisse im Plattformbetrieb frühzeitig zu erkennen und Reaktionen gezielt zu priorisieren. Tabelle 4.4 fasst das Muster als Plattformaufgabe zusammen. Im Fokus stehen dabei sowohl die Priorisierung von Sicherheits-Scans als auch die laufende Auswertung von Log- und Netzwerkdaten. Auf dieser Basis lassen sich sicherheitsrelevante Abweichungen früh erkennen und automatisierte Maßnahmen einleiten, etwa die gezielte Isolation betroffener Komponenten.

Tabelle 4.4: Übersicht zum Muster DevSecOps (Security Ops)

DevSecOps (Security Ops)	**Beschreibung**
Konkreter Plattform-Task	Priorisierung von Sicherheits-Scans sowie Echtzeit-Erkennung von Angriffen (z.B. DDoS, Malware) anhand von Netzwerk- und Audit-Telemetrie.
Empfohlene KI-Methode	Deep Learning (z.B. CNNs auf flow-basierten Merkmalsrepräsentationen) oder Gradient Boosting (z.B. XGBoost) für unausgewogene Sicherheitsdaten [20].
KPI/Mehrwert	Erkennungsgenauigkeit bis ca. 95 %; Reduktion von Fehlalarmen um ca. 20 %; hohe Detektionsraten auch für neuartige Angriffsmuster (studienabhängig) [20].
Limitationen	Hoher Rechenbedarf (insbesondere bei Deep-Learning-Inferenz); Datenschutz- und Compliance-Risiken (DSGVO) bei der Analyse paketbasierter Protokolle und potenziell personenbezogener Daten.
Einsatzvoraussetzungen	Echtzeit-Erfassung von Netzwerk-Flows (z.B. NetFlow/sFlow) und Logdaten (z.B. Cloud-Audit-Logs) sowie Integration von Threat-Intelligence-Informationen (z.B. MITRE ATT&CK) zur Kontextualisierung und Priorisierung.
Konkrete Tools	IBM Watson, Microsoft Sentinel, Amazon GuardDuty, Trivy, Grype, Snyk, Falco, Aqua Security, Kyverno, Clair [12, 1, 30]

Als methodischer Ansatz werden in der Literatur sowohl Deep-Learning-Modelle (z.B. CNNs) als auch Gradient-Boosting-Verfahren (z.B. XGBoost) eingesetzt [27, 20]. XGBoost kann dabei besonders dann stabil funktionieren, wenn es nur wenige echte Sicherheitsvorfälle in den Daten gibt [27, 20]. Der Nutzen im Betrieb liegt in einer besseren Erkennung bei weniger Fehlalarmen. Dadurch verringert sich der manuelle Aufwand, und kritische Funde können schneller weiterbearbeitet werden [20]. Gleichzeitig kann der Betrieb aufwendig werden, weil Deep-Learning-Modelle beim Einsatz zusätzliche Rechenleistung benötigen und damit Kosten verursachen. Zusätzlich stei-

gen die Anforderungen an Datenschutz und Compliance, sobald Log- oder Netzwerkdaten personenbezogene Informationen enthalten und nach DSGVO verarbeitet werden müssen. Für einen stabilen Einsatz ist daher eine verlässliche Erfassung von Netzwerk- und Logdaten erforderlich. Diese Daten müssen so zusammengeführt werden, dass sicherheitsrelevante Ereignisse nachvollziehbar eingeordnet und automatisierte Reaktionen gezielt und kontrolliert ausgelöst werden können.

5

Anwendung der Literaturrecherche

In diesem Kapitel wird das in Kapitel 4 abgeleitete Konzept zur Bewertung und Priorisierung von KI-Use-Cases angewendet. Abschnitt 5.1 beschreibt das Bewertungskonzept, Abschnitt 5.2 ordnet die Bosch Digital Manufacturing Plattform als Anwendungskontext ein. In Abschnitt 5.3 wird das Konzept exemplarisch auf ein konkretes Problemfeld angewendet und in Abschnitt 5.4 wird daraus eine Handlungsempfehlung abgeleitet.

5.1. Bewertungskonzept

Aufbauend auf den zuvor identifizierten KI-Use-Cases wird im Folgenden ein Bewertungskonzept vorgestellt, das eine strukturierte Einordnung und Vergleichbarkeit dieser Anwendungsfälle ermöglicht. Ziel des Frameworks ist es, KI-Anwendungsfälle im Cloud-Native Platform Engineering systematisch hinsichtlich ihres Implementierungsaufwands und ihres operativen Mehrwerts zu bewerten. Damit soll eine fundierte Entscheidungsgrundlage für die Priorisierung und Implementierung von KI-Lösungen geschaffen werden.

5.1.1. Ziel und Einordnung der Bewertungslogik

Der Implementierungsaufwand beschreibt den Aufwand, eine KI-Funktion robust, sicher und wartbar in den Betrieb zu integrieren. Der operative Mehrwert beschreibt den erwarteten Beitrag zur messbaren Verbesserung zentraler Betriebsziele. Dazu zählen insbesondere Betriebszuverlässigkeit (z. B. MTTR), Effizienz und Systemstabilität. Aspek-

te wie Datenzugang, Steuerung und organisatorische Voraussetzungen werden nicht in der X/Y-Achse abgebildet, sondern über zusätzliche Dimensionen betrachtet.
Die X-Achse bildet den Implementierungsaufwand ab. Tabelle 5.1 beschreibt die Ausprägungen von niedrig bis hoch. Niedrig bedeutet, dass sich der Use Case mit vorhandenen Funktionen oder Standardwerkzeugen umsetzen lässt und der Integrationsaufwand gering bleibt. Hoch bedeutet, dass zusätzliche Entwicklung, tiefere Integration und ein spürbarer Betriebsaufwand notwendig sind.

Tabelle 5.1: X-Achse: Implementierungsaufwand

Ausprägung	**Beschreibung**
Niedrig	Einsatz vorhandener KI-Funktionen oder Standardtools ohne eigenes Modelltraining
Mittel	Anpassung bestehender Modelle, Feature Engineering, begrenztes Retraining
Hoch	Eigenes Modelltraining, kontinuierlicher Betrieb und Überwachung notwendig

Die Y-Achse bildet den operativen Mehrwert ab. Tabelle 5.2 konkretisiert auch hier die Ausprägungen von niedrig bis hoch. Niedrig steht für unterstützende Funktionen, die zentrale Betriebsgrößen nur gering beeinflussen. Hoch liegt vor, wenn sich messbare Verbesserungen zeigen, etwa bei MTTR, Infrastrukturkosten oder der Systemstabilität.

Tabelle 5.2: Y-Achse: Operativer Mehrwert

Ausprägung	**Beschreibung**
Niedrig	Geringer Einfluss auf den Plattformbetrieb, primär unterstützende Funktionen wie Log-Analyse oder einfache Anomalieerkennung.
Mittel	Messbare Effizienzsteigerung oder Zeitersparnis im Plattformbetrieb durch KI-gestützte Automatisierung oder Optimierung.
Hoch	Deutliche Verbesserung zentraler KPIs (z.B. MTTR, Kostenreduktion, Stabilität)

Abbildung 5.1 führt beide Dimensionen in einer Bewertungsmatrix zusammen. Die Matrix ist in vier Quadranten unterteilt, wodurch sich Anwendungsfälle qualitativ nach Aufwand und Mehrwert einordnen lassen. Anwendungsfälle mit hohem Mehrwert und niedrigem Aufwand eignen sich für eine frühe Umsetzung. Use Cases mit hohem Mehrwert und hohem Aufwand bleiben relevant, erfordern jedoch typischerweise Vorarbeiten, etwa bei Daten- und Integrationsgrundlagen. Anwendungsfälle mit hohem Aufwand und

geringem Mehrwert sollten nur weiterverfolgt werden, wenn klare Abhängigkeiten bestehen.

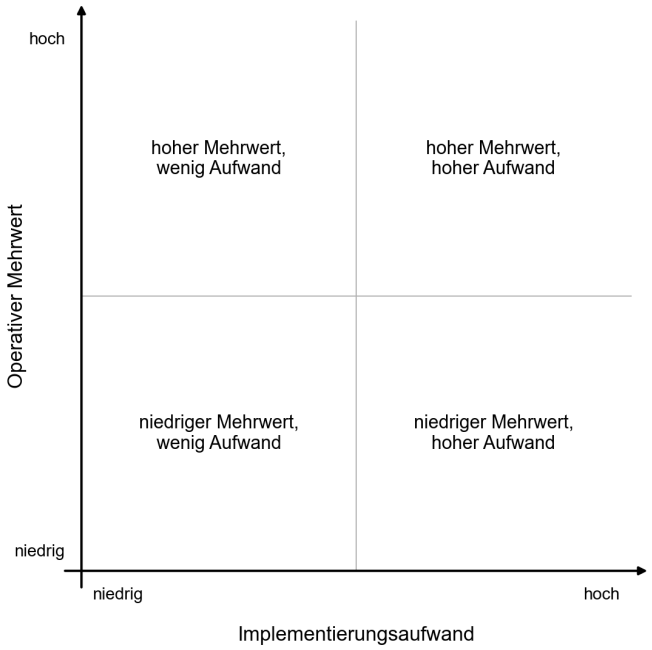


Abbildung 5.1: Bewertungsmatrix entlang von Implementierungsaufwand und operativem Mehrwert

5.1.2. Zusatzdimensionen und Bewertungsstufen

Die zweidimensionale Bewertungsmatrix priorisiert KI-Use-Cases zunächst über das Verhältnis von Implementierungsaufwand (X) und operativem Mehrwert (Y). Sie liefert damit eine erste Einordnung, greift aber zentrale Rahmenbedingungen des Plattformbetriebs nur indirekt auf.
Daher werden ergänzend vier Zusatzdimensionen betrachtet. Diese dienen als Bewertungsgrundlage, um Implementierungsaufwand (X) und operativen Mehrwert (Y) nachvollziehbar abzuleiten. Die Bewertung erfolgt je Unterdimension einheitlich auf einer dreistufigen Skala (niedrig/mittel/hoch). Die Zusatzdimensionen sind nicht als separate Achsen dargestellt, sondern bilden die Grundlage zur Ableitung der X- und Y-Werte und damit zur Positionierung in der Matrix.

Dimension 1: Umsetzbarkeit
Die Dimension Umsetzbarkeit bewertet, wie realistisch ein KI-Use-Case technisch und organisatorisch umgesetzt werden kann. Sie umfasst (i) Datenverfügbarkeit und -qualität, (ii) die Passung zur bestehenden Plattform- bzw. CI/CD-Architektur (inkl. Integration) sowie (iii) die verfügbaren Fachkenntnisse für Entwicklung, Betrieb und Wartung.
Niedrig bedeutet, dass Daten ausreichend vorhanden sind, die Integration ohne größere

Änderungen möglich ist und die nötige Expertise im Team verfügbar ist.
Mittel bedeutet, dass einzelne Vorarbeiten nötig sind (z.B. Datenaufbereitung, kleinere Anpassungen in Schnittstellen/Prozessen oder gezielter Kompetenzaufbau).
Hoch bedeutet, dass wesentliche Voraussetzungen fehlen (z.B. Datenlücken, fehlende Integrationsmöglichkeiten oder deutlicher Kompetenzaufbau), sodass der Use Case nur mit spürbarem Vorlauf realisierbar ist.

Dimension 2: Betriebswirksamkeit & Skalierbarkeit
Diese Dimension bewertet, ob ein KI-Use-Case im laufenden Plattformbetrieb zuverlässig einen operativen Nutzen entfaltet. Zusätzlich wird betrachtet, ob sich der Ansatz mit vertretbarem Aufwand auf weitere Services oder Teams übertragen lässt. Berücksichtigt werden (i) die Stabilität und Wirksamkeit im Betrieb sowie (ii) die Skalierbarkeit und Wiederverwendbarkeit der Lösung.
Niedrig bedeutet, dass der Nutzen stark kontextabhängig, instabil oder auf einzelne Services begrenzt ist.
Mittel bedeutet, dass ein erkennbarer Mehrwert im Betrieb entsteht, die Übertragbarkeit jedoch nur eingeschränkt gegeben ist oder zusätzlichen Anpassungsaufwand erfordert.
Hoch bedeutet, dass die Lösung stabil im Betrieb wirkt und mit geringem Zusatzaufwand breit ausgerollt werden kann.

Dimension 3: Compliance & Governance
Diese Dimension erfasst organisatorische und regulatorische Rahmenbedingungen für den Einsatz eines KI-Use-Cases. Bewertet werden (i) Anforderungen an Datenschutz, Datenklassifikation und Zugriffskontrolle sowie (ii) die organisatorische Verankerung durch klare Zuständigkeiten und Freigabeprozesse. Die Dimension dient dazu, Risiken und Abstimmungsbedarfe frühzeitig sichtbar zu machen.
Niedrig bedeutet, dass geringe regulatorische Anforderungen bestehen und Verantwortlichkeiten klar geregelt sind.
Mittel bedeutet, dass zusätzliche Abstimmungen oder formale Freigaben erforderlich sind, jedoch gut handhabbar.
Hoch bedeutet, dass strenge Vorgaben, sensible Daten oder unklare Zuständigkeiten zu erhöhtem Abstimmungs- und Umsetzungsaufwand führen.

Dimension 4: Reifegrad
Die Dimension Reifegrad bewertet den Entwicklungs- und Betriebsstand eines KI-Use-Cases. Sie umfasst (i) die Technologiereife der Lösung sowie (ii) die Absicherung des dauerhaften Betriebs. Berücksichtigt werden unter anderem der Entwicklungsstatus, vorhandene Betriebserfahrungen und die organisatorische Absicherung.
Niedrig bedeutet, dass der Use Case als Konzept oder Prototyp vorliegt und belastbare Erfahrungen im Regelbetrieb fehlen.

Mittel bedeutet, dass der Use Case pilotiert oder eingeschränkt produktiv eingesetzt wurde, mit noch offenen Betriebsfragen.
Hoch bedeutet, dass die Lösung produktiv im Einsatz ist und technisch sowie organisatorisch abgesichert betrieben wird.

Zur einheitlichen Anwendung der Zusatzdimensionen werden die qualitativen Bewertungen in Tabelle 5.3 zusammengefasst.

Tabelle 5.3: Zusatzdimensionen der Bewertung von KI-Use-Cases im Plattformbetrieb

Zusatzdimension	Unterdimension	Ziel / Leitfrage	Achse	Gewichtung	Quelle
Umsetzbarkeit	Datenverfügbarkeit und -qualität	Sind ausreichend vollständige und historisch nutzbare Daten für Training und Betrieb vorhanden?	X	2	[12, 8, 18]
	Technischer Fit	Lässt sich der Use Case realistisch in bestehende Plattform-, oder CI/CD-Architekturen integrieren?	X	2	[15, 14]
	Erforderliche Fachkenntnisse	Ist ausreichende Expertise für Entwicklung, Betrieb und Wartung KI-gestützter Funktionen vorhanden?	X	1	[18, 8]
Betriebswirksamkeit & Skalierbarkeit	Stabilität und operativer Nutzen	Erzielt die Lösung im laufenden Plattformbetrieb einen stabilen, messbaren Mehrwert?	Y	2	[15, 16, 14]
	Skalierbarkeit und Wiederverwendbarkeit	Lässt sich der Ansatz ohne hohen Individualaufwand auf weitere Services oder Teams übertragen?	Y	1	[8, 12]
Compliance & Governance	Datenschutz und Datenzugriff	Welche Anforderungen ergeben sich aus Datenschutz, Datenklassifikation und Zugriffskontrolle?	X	2	[12, 20]
	Organisatorische Verankerung	Sind Verantwortlichkeiten, Freigaben und Governance-Strukturen klar definiert und akzeptiert?	X	1	[8, 13]
Reifegrad	Technologiereife	In welchem Stadium befindet sich die Lösung (Konzept, Pilot, Produktivbetrieb)?	X+Y	1	[27, 13]
	Betriebsreife	Ist der dauerhafte Betrieb durch klare Zuständigkeiten, Überwachungsmechanismen und eine ausreichende Betriebsdokumentation abgesichert?	X+Y	1	[12, 15, 14]

5.1.3. Ableitung der X- und Y-Achse und Quadrantenzuordnung

Die Einordnung eines KI-Use-Cases in der Bewertungsmatrix erfolgt auf Basis eines Bewertungsrasters. Das Raster umfasst die Unterdimensionen der vier Zusatzdimensionen aus Abschnitt 5.1.2. Für jede Unterdimension wird die zugehörige Leitfrage beantwortet und auf einer dreistufigen Skala (niedrig/mittel/hoch) bewertet, die auch in Tabelle 5.3 dargestellt ist. Zur Auswertung werden die Stufen in Zahlenwerte überführt (niedrig = 1, mittel = 2, hoch = 3).
Der Wert der X-Achse ergibt sich als Mittelwert aller Bewertungen, die der Achse X oder beiden Achsen zugeordnet sind. Analog wird der Wert der Y-Achse berechnet. Unterdimensionen, die beide Achsen betreffen, werden entsprechend in beiden Berechnungen berücksichtigt. Die Unterdimensionen können dabei unterschiedlich gewichtet werden, um ihre relative Bedeutung für Implementierungsaufwand bzw. operativen Mehrwert abzubilden. In der Excel-Vorlage erfolgt dies über einen gewichteten Mittelwert, bei dem Standardgewichtungen auf 1 gesetzt und besonders relevante Kriterien höher gewichtet werden.
Auf Basis der berechneten X- und Y-Werte wird ein Use Case in die Bewertungsmatrix eingeordnet. Zusätzlich wurde ein Schwellenwert definiert, um die Quadranten klar abzugrenzen. Als hoch gilt ein Achsenwert ab 2,34, um eine klare Abgrenzung zwischen niedriger und hoher Ausprägung sicherzustellen und Verzerrungen durch einzelne Extrembewertungen zu vermeiden. Der resultierende Quadrant dient als Grundlage für die weitere qualitative Einordnung und Interpretation im folgenden Abschnitt.
Aus den Achsenwerten ergibt sich eine Position in der Bewertungsmatrix, die in das X/Y-Diagramm übertragen wird. Da Mittelwerte verwendet werden, kann die Position auch zwischen Quadranten liegen. Dies bildet gemischte Bewertungen ab, etwa wenn einzelne Voraussetzungen erfüllt sind, andere jedoch noch Vorarbeiten erfordern.
Das Bewertungsraster ist als Excel-Vorlage umgesetzt und im Anhang dokumentiert. Es ermöglicht eine einheitliche, nachvollziehbare und reproduzierbare Einordnung der Use Cases.

5.2. Analyse der Bosch Digital Manufacturing Plattform

Die Bosch Digital Manufacturing Plattform (BMLP) ist eine modular aufgebaute, Cloud-Native-Plattform zur Unterstützung von Fertigungs- und Logistikprozessen. Anwendungen werden als lose gekoppelte Services bereitgestellt und in verteilten Umgebungen betrieben. Dazu zählen Edge-Standorte in den Werken, zentrale Rechenzentren sowie cloudbasierte Instanzen. Die Plattform ist auf einen hybriden Betrieb ausgelegt und adressiert Anforderungen an Latenz, Verfügbarkeit und Compliance gleichermaßen.
Cloud-Native ist die Plattform, da sie konsequent auf Container-Orchestrierung mit Kubernetes, deklarative Schnittstellen, automatisierte CI/CD-Prozesse sowie durchgängige Beobachtbarkeit setzt. Diese Eigenschaften bilden die technische Grundlage für eine hohe Änderungsfrequenz und einen standardisierten Plattformbetrieb. Gleichzeitig entstehen umfangreiche Betriebsdaten, die für KI-gestützte Verfahren im Plattformbetrieb nutzbar sind. Alle Informationen basierend auf internen Plattformdokumentationen und Expertengesprächen (nicht öffentlich zugänglich).

5.2.1. Architektur und Betriebsmodell

Fachliche Funktionen in der BMLP werden als eigenständige deploybare Module umgesetzt, auf Basis der verteilten und modularen Grundstruktur. Jedes Modul bringt typischerweise eigene Logik und eigene Datenhaltung mit, wodurch direkte Abhängigkeiten zwischen Modulen vermieden werden. Die Zusammenarbeit zwischen Modulen erfolgt überwiegend ereignisbasiert, während synchrone Schnittstellen nur für spezielle Anwendungsfälle genutzt werden.
Für den werks- und organisationsübergreifenden Einsatz unterstützt die Plattform eine klare Trennung von Zugriffen und Verantwortlichkeiten. Diese wird durch zentrale Plattformdienste und ein gemeinsames Rollen- und Berechtigungsmodell umgesetzt. Dadurch können mehrere Organisationseinheiten dieselbe Plattform nutzen, ohne die Zugriffskontrolle zu verlieren.
Im Betrieb werden zentrale Mechanismen möglichst vereinheitlicht. Dazu zählen gemeinsame Einstiegspunkte für Anwendungen sowie einheitliche Informationen zum Betriebszustand der Module. Monitoring und Supportprozesse bauen auf diesen Grundlagen auf und sind über die Plattform hinweg vergleichbar organisiert.

5.2.2. Entwicklungsmodell und Plattformstandards

Die Entwicklung der Module erfolgt intern und ist über mehrere Repositories organisiert. Zentrale Entwicklungsschritte wie Build, Tests, Container-Erstellung und Sicherheitsprüfungen sind in standardisierten CI-Pipelines automatisiert. Die daraus entstehenden Artefakte werden versioniert abgelegt und für den weiteren Betrieb bereitgestellt.
Auch die Auslieferung folgt einem weitgehend automatisierten Ansatz. Ein wiederverwendbares Pipeline-Modell verbindet Artefaktverwaltung, Konfigurations- und Geheimnisverwaltung sowie deklarative Deployments zu einem durchgängigen Prozess. Ziel ist eine reproduzierbare und nachvollziehbare Bereitstellung, die zugleich regulatorische Anforderungen berücksichtigt.
Ergänzend definieren Plattformstandards zentrale Leitplanken für Entwicklung und Betrieb. Dazu zählen einheitliche Vorgaben für Container-Images und deklarative Infrastruktur. Schnittstellen werden nach gemeinsamen Gestaltungsregeln umgesetzt und konsistent dokumentiert, während die Kommunikation zwischen Modulen überwiegend ereignisbasiert erfolgt. Für den Betrieb werden einheitliche Logformate und durchgängige Korrelationskennungen genutzt, sodass Abläufe über mehrere Komponenten hinweg nachvollziehbar bleiben.

5.2.3. Operativer Stack und Datenbasis für AIOps

Für AIOps ist weniger die Tool-Landschaft entscheidend als die Daten, die im Plattformbetrieb entstehen. In der BMLP werden Betriebsdaten über Logs, Metriken, Traces sowie technische und fachliche Ereignisse erfasst und zentral nutzbar gemacht. Logs liegen in einem einheitlichen Format vor und enthalten neben Zeitstempel und Systemkontext auch Versions- und Instanzinformationen sowie Korrelationskennungen. Damit lassen sich Abläufe über mehrere Komponenten hinweg zusammenführen und automatisiert auswerten.
Metriken und Traces ergänzen diese Sicht um messbare Größen wie Last, Latenz und Fehlerraten sowie um Ablaufspuren über Servicegrenzen hinweg. Ereignisdaten bilden Zustandswechsel und Prozessketten ab und unterstützen dadurch die Einordnung von Störungen. Zusätzlich entstehen Daten aus Build- und Rollout-Prozessen, etwa Testergebnisse, Sicherheitsprüfungen und Rollout-Verläufe, die Rückschlüsse auf Änderungsrisiken erlauben. Auch Änderungen an Konfigurationen können Hinweise auf Abweichungen zwischen geplantem und aktuellem Zustand liefern.
Auf dieser Datenbasis lassen sich insbesondere Auffälligkeiten frühzeitig erkennen und Störungen schneller eingrenzen. Ebenso können Rollouts datenbasiert überwacht und Kapazitätsbedarfe besser abgeschätzt werden.

5.3. Anwendung des Bewertungskonzepts

In diesem Abschnitt wird das in Kapitel 5.1 entwickelte Bewertungskonzept auf ein konkretes Problemfeld der Bosch Digital Manufacturing Plattform angewendet. Ziel ist es, dieses Problemfeld systematisch einzuordnen und anhand der definierten Zusatzdimensionen zu bewerten. Die Analyse folgt dabei den generischen Use-Case-Mustern aus Kapitel 4.3.

Im Betrieb der Bosch Digital Manufacturing Plattform treten Abbrüche in Build- und Bereitstellungsprozessen immer wieder auf. Sie entstehen im Rahmen automatisierter CI/CD-Pipelines und betreffen sowohl einzelne Module als auch verkettete Abläufe über mehrere Komponenten hinweg. Auch wenn keine konsolidierten Kennzahlen zur Häufigkeit vorliegen, zeigen interne Analysen und Betriebserfahrungen, dass solche Abbrüche wiederkehrend auftreten und die Durchlaufzeiten von Änderungen spürbar verlängern. Die Auswirkungen sind insbesondere in Umgebungen mit paralleler Entwicklung und hoher Änderungsfrequenz sichtbar, da fehlerhafte Builds nachgelagerte Schritte blockieren und erneute Pipeline-Läufe erforderlich machen. Dadurch verschiebt sich die Bereitstellung von Funktionen, was die Geschwindigkeit der Änderungsbereitstellung der Plattform insgesamt beeinträchtigt.

Der zentrale Aufwand entsteht nicht durch den Abbruch eines Builds an sich, sondern durch die anschließende Ursachenanalyse. Fehlersituationen lassen sich häufig nicht eindeutig einem einzelnen Schritt oder einer einzelnen Komponente zuordnen. Stattdessen müssen Informationen aus mehreren Quellen zusammengeführt werden, darunter Pipeline-Läufe, System- und Applikationsprotokolle sowie technische Ereignisse aus der Laufzeitumgebung. Eine durchgängige Sicht über den gesamten Ablauf ist dabei nur eingeschränkt gegeben. Detailinformationen stehen nicht dauerhaft zur Verfügung, sondern müssen bei Bedarf gezielt aktiviert werden. Dies erhöht den manuellen Analyseaufwand und verlangsamt die Eingrenzung der eigentlichen Ursache.
Zusätzlich zeigt sich, dass Build- und Bereitstellungsprozesse in der Praxis nicht vollständig unabhängig voneinander sind. Entlang der Build- und Auslieferungsprozesse bestehen Abhängigkeiten, etwa durch gemeinsam genutzte Artefakte, feste Reihenfolgen bei der Initialisierung oder gegenseitige Startvoraussetzungen einzelner Komponenten. Obwohl eine möglichst unabhängige Ausführung einzelner Pipelines angestrebt wird, führen diese Kopplungen dazu, dass Fehlerfolgen über mehrere Schritte hinweg wirken. Die Analyse erfordert dadurch ein Verständnis des Gesamtzusammenhangs und einen hohen manuellen Analyseaufwand für die beteiligten Teams.

Für die systematische Einordnung dieses Problemfelds liegt in der Plattform eine geeig-

nete Datengrundlage vor. Pipeline-Laufinformationen und Build-Protokolle werden erfasst und für Auswertungen bereitgestellt. Applikations- und Prozessprotokolle werden zusammengeführt und folgen einem einheitlichen, strukturierten Format mit standardisierten Feldern. Dadurch lassen sich relevante Informationen einheitlich auswerten und vergleichen, ohne dass jede Analyse bei null beginnt.
Zudem ist eine historische Protokollierung vorgesehen. Die Einträge enthalten unter anderem Zeitstempel sowie Angaben zu Umgebung, Systemkontext, Version und Instanz. Pipeline-Ausführungen und Laufzeitereignisse lassen sich über gemeinsame Kennungen und Kontextinformationen miteinander in Beziehung setzen und im Nachgang nachvollziehen. Damit sind zentrale Voraussetzungen gegeben, um wiederkehrende Fehlermuster zu identifizieren. Die Ursachen lassen sich systematischer eingrenzen, statt rein manuell von Einzelfall zu Einzelfall vorzugehen.

Das beschriebene Problemfeld wird dem in Abschnitt 4.3.3 abgeleiteten generischen Use Case der intelligenten Build-Fehlerdiagnose zugeordnet. Auf dieser Grundlage erfolgt die Bewertung entlang der in Abschnitt 5.1.2 definierten Zusatzdimensionen.
Bezüglich der Umsetzbarkeit sind die erforderlichen Build-, Pipeline- und Protokolldaten in strukturierter Form und mit historischer Tiefe verfügbar. Auch der technische Fit zur bestehenden Plattform ist gegeben, da die Analyse an vorhandene Build- und Auslieferungsprozesse anknüpfen kann. Zusätzlicher Aufwand entsteht durch die Einbettung in bestehende Abläufe sowie durch den Umgang mit Abhängigkeiten zwischen Komponenten und Pipelines. Insgesamt ergibt sich eine mittlere Umsetzbarkeit.
Durch die Auswertung historischer Build- und Protokolldaten kann der manuelle Analyseaufwand reduziert und die Eingrenzung von Fehlerursachen beschleunigt werden. Eine Übertragung auf weitere Pipelines ist möglich, erfordert jedoch Anpassungen, wenn Abläufe eng miteinander verknüpft sind. In der Gesamtsicht ergibt sich eine mittlere bis hohe Ausprägung.
Da überwiegend technische Prozess- und Protokolldaten verarbeitet werden, entstehen keine zusätzlichen Anforderungen an Datenschutz oder Regulierung. Abstimmungsaufwände betreffen vor allem Zuständigkeiten und organisatorische Regelungen, nicht jedoch regulatorische Einschränkungen. Die Dimension ist daher als niedrig bis mittel einzuordnen.
Der Reifegrad des Use Case ist insgesamt als mittel einzuordnen. In der Literatur wird der Ansatz über die reine Konzeptionsphase hinaus beschrieben und in ersten Werkzeugen sowie Pilotanwendungen praktisch eingesetzt. Gleichzeitig fehlt jedoch bislang eine breite Etablierung mit klar abgesicherten Betriebsmodellen und Verantwortlichkeiten für den dauerhaften Einsatz.

Abbildung 5.2 zeigt die Einordnung des Use Case in die Bewertungsmatrix auf Basis

der in diesem Abschnitt durchgeführten Analyse und Anwendung des Bewertungsrasters.

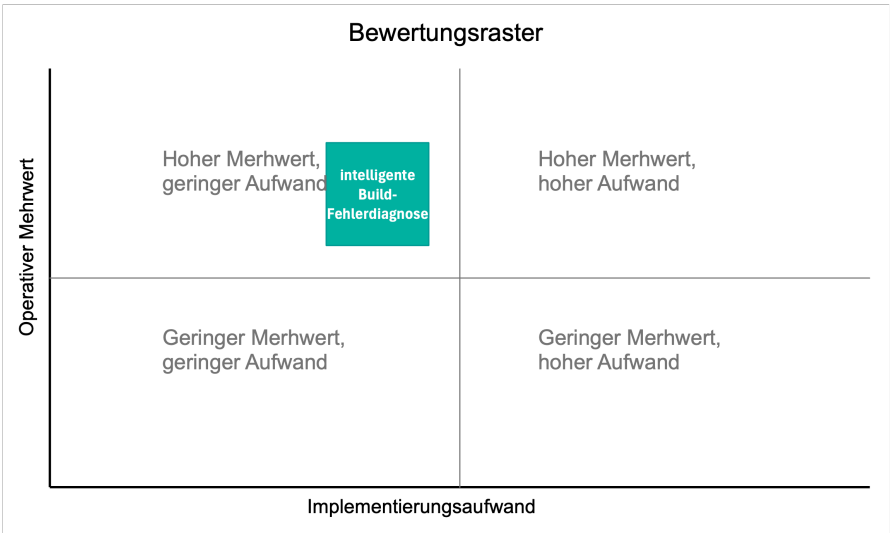


Abbildung 5.2: Einordnung des Use Case in die Bewertungsmatrix

5.4. Handlungsempfehlung

Auf Basis der Einordnung und Bewertung aus Abschnitt 5.3 wird in diesem Kapitel eine konkrete Handlungsempfehlung abgeleitet. Der Use Case der intelligenten Build-Fehlerdiagnose wurde im Bewertungsraster als hoch hinsichtlich des operativen Mehrwerts und mit mittlerer Umsetzbarkeit eingestuft. Daraus ergibt sich die Empfehlung, diesen Use Case priorisiert zu adressieren.

Auf Grundlage dieser Bewertung sollte die intelligente Build-Fehlerdiagnose als erster Anwendungsfall weiterverfolgt werden. Der erwartete Nutzen ergibt sich insbesondere aus der Reduktion des manuellen Analyseaufwands sowie einer schnelleren Eingrenzung von Fehlerursachen in Build- und Bereitstellungsprozessen. Gleichzeitig ist der erforderliche Aufwand überschaubar, da die notwendigen Datenquellen bereits vorhanden sind und keine grundlegenden Änderungen an bestehenden Abläufen erforderlich sind.
Eine priorisierte Umsetzung erlaubt es, das identifizierte Problemfeld gezielt zu adressieren und gleichzeitig Erfahrungen im Umgang mit datengetriebenen Analyseansätzen im Plattformbetrieb zu sammeln. Der Use Case eignet sich damit sowohl zur operativen Verbesserung als auch als Ausgangspunkt für weitere, darauf aufbauende Anwendungsfälle.

Auf Basis der priorisierten Einordnung bietet sich der Einsatz von K8sGPT als beispielhafte Umsetzung des identifizierten Use Case an [12]. In der Literatur wird K8sGPT als Open-Source-Projekt beschrieben, das den Einsatz von KI zur Unterstützung des Betriebs containerbasierter Systeme adressiert.
Der Schwerpunkt des Werkzeugs liegt auf der Analyse technischer Betriebsinformationen, insbesondere von Protokolldaten aus Kubernetes-Umgebungen. Ziel ist es, Betreiber bei der Ursachenanalyse von Fehlerzuständen zu unterstützen und wiederkehrende Probleme schneller nachvollziehbar zu machen. Damit greift der Ansatz direkt das in Abschnitt 5.3 beschriebene Problemfeld der Build- und Fehlerdiagnose auf.
Der Einsatz von K8sGPT ist als ergänzende Unterstützung bestehender Betriebs- und Analyseprozesse zu verstehen. Entscheidungen und Maßnahmen verbleiben weiterhin bei den verantwortlichen Teams. Das Werkzeug eignet sich somit, um den identifizierten Use Case technisch umzusetzen, ohne bestehende Abläufe grundlegend zu verändern.

Abbildung 5.3 zeigt eine einfache Architekturskizze, die die Einordnung von K8sGPT im Plattformbetrieb darstellt.

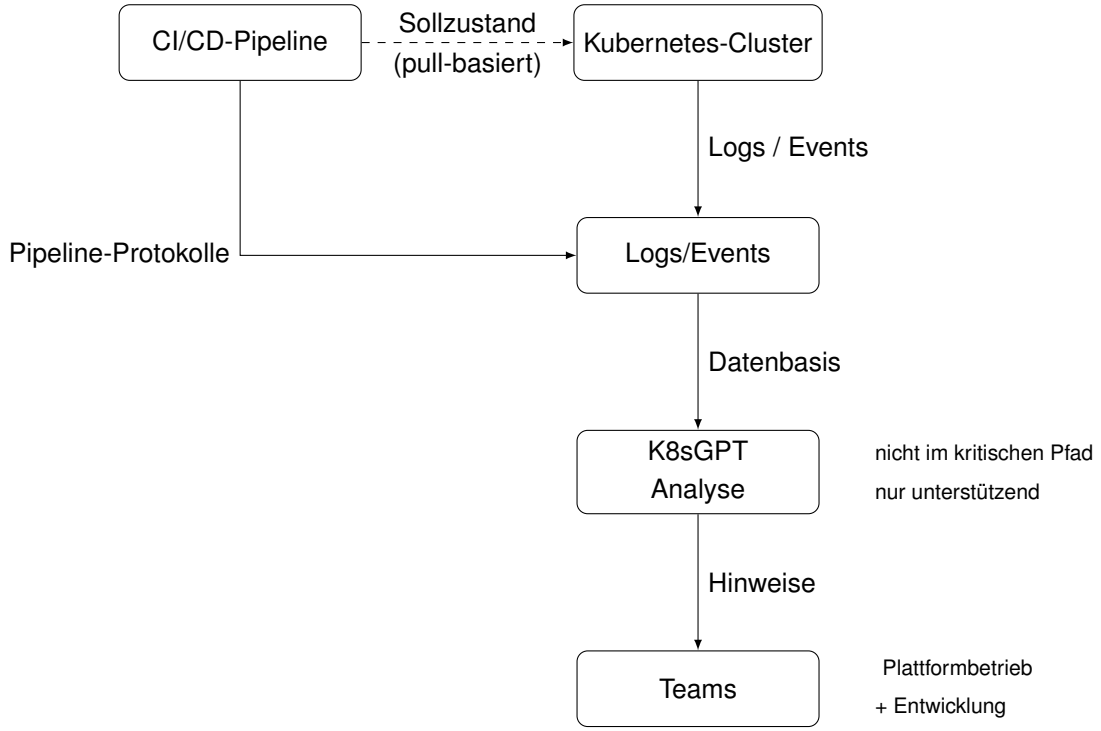


Abbildung 5.3: Einordnung von K8sGPT im Plattformbetrieb

Fehler entstehen entweder in der CI/CD-Pipeline oder in der Laufzeitumgebung des Kubernetes-Clusters. Die dabei anfallenden Logs, Ereignisse und Zustände werden zentral gesammelt und bilden die Datenbasis für die Analyse.
K8sGPT wertet diese Daten aus und generiert darauf basierende Hinweise für die zuständigen Teams. Das Werkzeug ist nicht in den Auslieferungsprozess eingebunden und hat keinen Einfluss auf Build oder Bereitstellung. Die Verantwortung für Entscheidungen und Maßnahmen verbleibt vollständig bei den beteiligten Teams.

Die Handlungsempfehlung beschreibt keinen vollständigen Zielzustand der Plattform. Sie zeigt lediglich, wie der identifizierte Use Case beispielhaft umgesetzt werden kann.

6

Diskussion

Die Ergebnisse werden eingeordnet, Limitationen diskutiert und Implikationen abgeleitet.

6.1. Beantwortung der Forschungsfragen

In diesem Abschnitt werden die drei Forschungsfragen der Arbeit (RQ1, RQ2, RQ3) auf Basis der erarbeiteten Ergebnisse beantwortet. Die Antworten beziehen sich auf die in der Literaturanalyse identifizierten Anwendungsfelder und Herausforderungen (RQ1), die untersuchten KI-Methoden und -Werkzeuge (RQ2) sowie das entwickelte Bewertungskonzept und Erkenntnisse aus Fallstudien (RQ3). Dadurch werden die in Kapitel 3.2 formulierten Fragen detailliert adressiert und die Erkenntnisse in den Gesamtkontext des Cloud-Native Platform Engineerings eingeordnet.

6.1.1. Beantwortung der Forschungsfrage RQ1

Die Literaturanalyse zeigt vier wiederkehrende Anwendungsfelder, in denen KI einen Mehrwert bringt und gleichzeitig auch Herausforderungen mit sich bringt. Diese lassen sich entlang konkreter betrieblicher Aufgaben abgrenzen: (1) Ressourcen- und Workload-Optimierung, (2) Betrieb und Orchestrierung der Plattform, (3) Optimierung von CI/CD-Pipelines sowie (4) Sicherheits- und Bedrohungserkennung.
Das Proaktiven Ressourcen-Management thematisiert die Skalierung von Workloads, zum Beispiel um die Anzahl von Kubernetes-Pods bei schwankender Last. Die KI wertet

Auslastungs- und Latenzverläufe aus und leitet daraus eine Vorhersage für die nächste Lastphase ab. Auf Basis dieser Vorhersage wird eine Skalierungsentscheidung früher getroffen als bei fester Zuteilung. Dadurch sinkt das Risiko, dass Services zu spät skalieren und unter Last instabil werden. Gleichzeitig wird eine Überbereitstellung reduziert, weil nicht zu viel Kapazität vorgehalten wird. Typische Herausforderungen entstehen, wenn nicht ausreichend historische Daten vorliegen oder Lastmuster sich häufig ändern.
Betrieb und Orchestrierung adressiert Störungen im laufenden Betrieb und die Eingrenzung ihrer Ursachen über mehrere Komponenten hinweg. Die Grundlage bilden Protokolle, Metriken und Ereignisse, die zu einen Hinweis geben, welche Komponente wahrscheinlich der Auslöser ist. Ein typisches Beispiel ist die Auswertung von Kubernetes-Events und Protokollen, um Fehlkonfigurationen oder wiederkehrende Fehlerbilder schneller zu erkennen. Der Mehrwert liegt in kürzerer Fehlersuche und einer schnelleren Einordnung, was wirklich relevant ist. Der Ansatz wird unzuverlässig, wenn sich das Normalverhalten durch Releases und Konfigurationsänderungen ständig verschiebt und dadurch viele Fehlalarme entstehen.
Die Absicherung von Ausrollungen zielt darauf ab, fehlerhafte Versionen früh zu begrenzen. Während einer schrittweisen Ausrollung werden Fehlerraten und Antwortzeiten beobachtet und mit dem erwarteten Normalzustand verglichen. Bei auffälligen Abweichungen kann eine Ausrollung gestoppt oder auf die vorige Version zurückgesetzt werden. Das ist relevant, weil Ausrollungen regelmäßig sind und Fehler sonst erst sichtbar werden, wenn sie bereits breiter wirken. Schwierig wird es, wenn normale Schwankungen nicht sauber von echten Fehlern getrennt werden und dadurch unnötige Rücksetzungen ausgelöst werden.
Sicherheits- und Bedrohungserkennung umfasst die Anomalieerkennung, Echtzeiterkennung von Angriffen sowie die Klassifikation von Bedrohungen. Ein greifbares Beispiel ist der Einsatz von SL-Algorithmen um erkannte Anomaliens als bösaritg oder gutartig einzustufen. Ergänzend können Laufzeitdaten Hinweise liefern, ob ein Fund in der konkreten Nutzung überhaupt wirksam wird. Dadurch fließt weniger Zeit in unkritische Funde und relevante Punkte können schneller bearbeitet werden. Grenzen entstehen durch sensible Daten und Vorgaben, weil Protokolle und Sicherheitsdaten nicht beliebig ausgewertet und aufbewahrt werden dürfen.
Über alle Felder hinweg zeigen sich wiederkehrende Herausforderungen. Dazu zählen zusätzlicher Rechen- und Betriebsaufwand, Integrationsaufwand in bestehende Abläufe sowie Einschränkungen durch Datenschutz und Freigaben. Am häufigsten limitiert jedoch die Datenbasis, weil unvollständige oder uneinheitliche Daten direkt zu instabilen Ergebnissen führen.

6.1.2. Beantwortung der Forschungsfrage RQ2

Die Literatur zeigt kein einheitliches Lernparadigma, Werkzeug oder Algorithmen, die für alle Anwendungsfälle gleichermaßen geeignet sind. Die Wahl hängt vor allem vom Anwendungsfall, von der verfügbaren Datenbasis und von der Systemarchitektur der Plattform ab.
Am häufigsten werden Supervised Learning Verfahren thematisiert und eingesetzt. Sie passen dort, wo historische Daten mit bekannten Ergebnissen vorliegen, zum Beispiel erfolgreiche und fehlgeschlagene Builds oder klassifizierte Störungen. Damit lassen sich Vorhersagen für Lastverläufe erstellen oder Fehlertypen aus Protokollen einordnen. Auch die Bewertung von Änderungsrisiken in CI/CD-Pipelines wird so umgesetzt, indem frühere Pipeline-Verläufe als Trainingsgrundlage dienen.
Unsupervised Learning Verfahren werden dementsprechend dort eingesetzt, wo solche Zuordnungen fehlen. Typisch ist die Erkennung von Abweichungen in Metriken oder Protokollen, ohne dass vorher markiert wurde, was ein Fehler ist. Das ist im Betrieb hilfreich, führt aber in dynamischen Umgebungen oft zu Fehlalarmen, weil sich das Normalverhalten durch Releases und Konfigurationsänderungen verschiebt.
Etwas weniger thematisiert aber trotzdem relevant sind Reinforcement Learning Ansätze. Diese werden vor allem für Steuerungsentscheidungen beschrieben, zum Beispiel für Skalierung oder Rollout-Entscheidungen unter wechselnden Bedingungen. Der Aufwand ist hoch, weil Training und Rückkopplung sauber abgebildet werden müssen, daher ist der Einsatz in produktiven Umgebungen bislang begrenzt.
Bei den Methoden dominieren komplexe Modelle, besonders neuronale Netze. Das liegt daran, dass sie mit heterogenen Betriebsdaten umgehen können, zum Beispiel mit Protokollen, Metriken und Ereignissen. Daneben werden auch klassische Verfahren genutzt, etwa baumbasierte Modelle für Klassifikation und Prognosen, oder Clustering für die Gruppierung ähnlicher Fehlerbilder. In mehreren Arbeiten wird deutlich, dass einfachere und besser nachvollziehbare Verfahren in manchen Fällen ausreichen würden, in der Forschung aber seltener im Fokus stehen.
Bei den Werkzeugen zeigt sich ein ähnliches Bild. Viele Ansätze bauen auf bestehenden Plattform- und Betriebswerkzeugen auf und ergänzen diese um KI-Funktionen. Im Kubernetes-Umfeld betrifft das zum Beispiel Autoscaling und die Auswertung von Events und Protokollen. Im CI/CD-Kontext werden Werkzeuge wie K8sGPT, Jenkins X oder DeepCode genannt, um Build- und Analyseaufgaben zu unterstützen. Für Rollout-Absicherung und Betrieb werden auch Plattformen wie Spinnaker oder Dynatrace beschrieben, die Daten aus dem Betrieb auswerten und Hinweise ableiten. Im Sicherheitskontext werden Werkzeuge wie Trivy, Snyk oder Falco genannt, die Funde aus Scans und Laufzeitdaten priorisieren.

Insgesamt zeigen die Ergebnisse, dass Verfahren und Werkzeuge nur dann sinnvoll einzuordnen sind, wenn der konkrete Use Case und die zugrunde liegende Datenbasis mitbetrachtet werden.

6.1.3. Beantwortung der Forschungsfrage RQ3

Die Forschungsfrage RQ3 untersucht, wie sich identifizierte KI-Lösungen auf typische Anwendungsfälle im Platform Engineering übertragen und hinsichtlich ihres Mehrwerts und ihrer Umsetzbarkeit bewerten lassen. Die Ergebnisse dieser Arbeit zeigen, dass eine strukturierte Bewertung möglich ist, wenn technische Ansätze nicht isoliert betrachtet, sondern in einen betrieblichen Kontext eingeordnet werden.
Als Grundlage dient das in Kapitel 5.1 entwickelte Bewertungskonzept, das zwei zentrale Dimensionen kombiniert. Die erste Dimension beschreibt den Implementierungsaufwand und wird auf der X-Achse abgebildet. Sie reicht von niedrigem bis hohem Aufwand und orientiert sich daran, in welchem Umfang bestehende Funktionen genutzt werden können, ob Anpassungen erforderlich sind oder ob ein eigenständiger Aufbau und kontinuierlicher Betrieb notwendig sind. Ein niedriger Implementierungsaufwand liegt vor, wenn vorhandene Werkzeuge oder Funktionen ohne eigenes Modelltraining eingesetzt werden können. Ein hoher Aufwand ist gegeben, wenn umfangreiche Integration, eigene Modellanpassungen sowie laufende Überwachung und Wartung erforderlich sind.
Die zweite Dimension bildet den operativen Mehrwert ab und wird auf der Y-Achse dargestellt. Auch hier erfolgt die Einordnung von niedrig bis hoch. Maßgeblich ist, in welchem Umfang ein Use Case zur messbaren Verbesserung zentraler Betriebsziele beiträgt. Ein niedriger operativer Mehrwert liegt vor, wenn KI lediglich unterstützende Funktionen erfüllt. Ein hoher Mehrwert ist gegeben, wenn sich deutliche Verbesserungen in Bereichen wie Stabilität, Reaktionszeiten oder Ressourceneffizienz erwarten lassen.
Zur Ergänzung dieser zweidimensionalen Einordnung wurden in Kapitel 5.1 zusätzliche Bewertungsdimensionen eingeführt. Die Dimension Umsetzbarkeit betrachtet, ob die notwendigen Datenquellen verfügbar sind, wie gut sich der Use Case in bestehende Plattformarchitekturen integrieren lässt und welches fachliche Know-how erforderlich ist. Die Dimension Betriebswirksamkeit und Skalierbarkeit bewertet, ob der erwartete Nutzen im laufenden Betrieb stabil erzielt werden kann und ob sich der Ansatz über mehrere Dienste oder Teams hinweg übertragen lässt. Die Dimension Governance und Reifegrad berücksichtigt Anforderungen an Datenschutz, Sicherheit und Nachvollziehbarkeit sowie den Entwicklungsstand der jeweiligen Lösung.
Die Bewertung entlang aller Dimensionen erfolgt jeweils qualitativ in den Stufen nied-

rig, mittel und hoch. Diese Einordnung orientiert sich nicht an einzelnen Technologien, sondern an strukturellen Kriterien wie Datenverfügbarkeit, Integrationsaufwand und betrieblicher Wirkung. Dadurch wird eine vergleichbare Bewertung unterschiedlicher Anwendungsfälle ermöglicht, ohne konkrete Umsetzungen vorwegzunehmen.
Insgesamt zeigt sich, dass die Übertragbarkeit von KI-Lösungen im Platform Engineering weniger von einzelnen Methoden abhängt als von einer konsistenten Betrachtung von Aufwand, Nutzen und betrieblichen Rahmenbedingungen. Das Bewertungskonzept ermöglicht es, generische Use-Case-Muster aus der Literatur systematisch auf konkrete Plattformkontexte zu beziehen und fundierte Priorisierungsentscheidungen vorzubereiten. Damit beantwortet die Arbeit die Forschungsfrage RQ3 auf einer strukturellen Ebene und schafft eine Brücke zwischen den Ergebnissen der Literaturanalyse und ihrer praktischen Einordnung.

6.2. Limitationen

Diese Arbeit basiert auf einer systematischen Literaturanalyse und einer darauf aufbauenden Anwendung auf eine konkrete Plattformumgebung. Daraus ergeben sich Grenzen, die bei der Einordnung der Ergebnisse zu beachten sind.
Die Literaturanalyse deckt nicht das gesamte Themenfeld ab. Als Ausgangspunkt wurden vier Basispublikationen genutzt und über eine Vorwärts- und Rückwärtssuche erweitert. Dadurch können relevante Arbeiten fehlen, etwa wenn sie nicht in den Zitationspfaden dieser Startmenge liegen oder andere Begriffe verwenden. Zusätzlich sind die Einordnung und Kategorisierung der Publikationen nicht vollständig objektiv. Auch bei klaren Kriterien bleibt eine gewisse Interpretationsabhängigkeit, zum Beispiel bei der Zuordnung zu Anwendungsfeldern oder bei der Ableitung der in Kapitel 4 beschriebenen Muster.
Die in Kapitel 4.3 abgeleiteten Use-Case-Muster sind bewusst generisch. Sie fassen wiederkehrende Aufgaben im Plattformbetrieb zusammen, bilden aber nicht alle Varianten und Randbedingungen ab, die in der Praxis auftreten. Das Bewertungskonzept in Kapitel 5 unterstützt eine strukturierte Priorisierung, bleibt jedoch qualitativ. Der tatsächliche Aufwand und der tatsächliche Nutzen hängen stark von der jeweiligen Datenlage, der Integrationsfähigkeit und den bestehenden Betriebsprozessen ab und wurden im Rahmen dieser Arbeit nicht durch Umsetzung oder Messungen validiert.
Die Anwendung des Bewertungskonzepts erfolgt anhand einer einzelnen Plattformumgebung und dient primär der Plausibilisierung des Vorgehens. Eine prototypische Implementierung war nicht Bestandteil der Arbeit, da insbesondere Datenqualität und Datenverfügbarkeit einen erheblichen Aufwand dargestellt hätten. Aussagen zu Effekten auf Kennzahlen wie Stabilität, Reaktionszeiten oder Kosten stellen daher begründete Einschätzungen dar und keine empirisch nachgewiesenen Ergebnisse.

7

Zusammenfassung und Ausblick

Abschließende Zusammenfassung der Arbeit sowie ein Ausblick auf zukünftige Forschung.

7.1. Zusammenfassung

Die wichtigsten Erkenntnisse und Ergebnisse der Arbeit werden hier zusammengefasst.

7.2. Ausblick

Mögliche zukünftige Forschungsrichtungen und offene Fragen werden hier diskutiert.

Tabellenverzeichnis

Abbildungsverzeichnis

Literaturverzeichnis

[1] Adel Zaalouk u. a. *CLOUD NATIVE ARTIFICIAL INTELLIGENCE*. Techn. Ber. 2024-03-20.

[2] *Overview*. https://kubernetes.io/docs/concepts/overview/. (Besucht am 15. 11. 2025).

[3] *GitOps in 2025: From Old-School Updates to the Modern Way*. https://www.cncf.io/blog/2025/06/09/g in-2025-from-old-school-updates-to-the-modern-way/. Juni 2025. (Besucht am 30. 12. 2025).

[4] juliakm. *What Is Platform Engineering?* https://learn.microsoft.com/en-us/platform-engineering/what-is-platform-engineering. (Besucht am 15. 12. 2025).

[5] *Was ist DevOps? Erläuterung zu DevOps — Microsoft Azure*. https://azure.microsoft.com/de-de/resources/cloud-computing-dictionary/what-is-devops. (Besucht am 16. 12. 2025).

[6] Atlassian. *Was ist DevOps?* https://www.atlassian.com/de/devops. (Besucht am 16. 12. 2025).

[7] Dhia Elhaq Rzig u. a. *Empirical Analysis on CI/CD Pipeline Evolution in Machine Learning Projects*. https://arxiv.org/abs/2403.12199v4. März 2024. (Besucht am 16. 12. 2025).

[8] Aliyu Enemosah. "Enhancing DevOps Efficiency through AI-Driven Predictive Models for Continuous Integration and Deployment Pipelines". In: *International Journal of Research Publication and Reviews* 6.1 (Jan. 2025), S. 871–887. ISSN: 25827421. DOI: 10.55248/gengpi.6.0125.0229. (Besucht am 03. 11. 2025).

[9] Dirk Valkenborg u. a. "Supervised Learning". In: *American Journal of Orthodontics and Dentofacial Orthopedics* 164.1 (Juli 2023), S. 146–149. ISSN: 08895406. DOI: 10.1016/j.ajodo.2023.04.010. (Besucht am 27. 11. 2025).

[10] Dirk Valkenborg u. a. "Unsupervised Learning". In: *American Journal of Orthodontics and Dentofacial Orthopedics* 163.6 (Juni 2023), S. 877–882. ISSN: 08895406. DOI: 10.1016/j.ajodo.2023.04.001. (Besucht am 27. 11. 2025).

[11] Majid Ghasemi u. a. *An Introduction to Reinforcement Learning: Fundamental Concepts and Practical Applications*. Aug. 2024. DOI: 10.48550/arXiv.2408.07712.

[12] Giridhar Kankanala und Sudheer Amgothu. "AI/ML – DevOps Automation". In: 13 (Okt. 2024), S. 111–117.

[13] Suprit Pattanayak, Pranav Murthy und Aditya Mehra. “Integrating AI into DevOps Pipelines: Continuous Integration, Continuous Delivery, and Automation in Infrastructural Management: Projections for Future”. In: *International Journal of Science and Research Archive* 13.1 (Okt. 2024), S. 2244–2256. ISSN: 25828185. DOI: 10.30574/ijsra.2024.13.1.1838. (Besucht am 03. 11. 2025).

[14] Varun Tamminedi. “Automating Kubernetes Operations with AI and Machine Learning”. In: *IJFMR - International Journal For Multidisciplinary Research* 6.6 (Dez. 2024). ISSN: 2582-2160. DOI: 10.36948/ijfmr.2024.v06i06.33430. (Besucht am 03. 11. 2025).

[15] Sudip Poudel u. a. “AI-Driven Intelligent Auto-Scaling for Cloud Resource Optimization 1”. In: *Journal of Advanced College of Engineering and Management* Vol. 11 (Okt. 2025), S. 27–36. DOI: 10.3126/jacem.v11i1.84521.

[16] Josson Paul Kalapparambath, Yugandhar Suthari und Gaurav Mishra. “Advancing Distributed Systems with Reinforcement Learning: A New Frontier in AI-Integrated Software Engineering”. In: (März 2025). ISSN: 2945-3585. DOI: 10.5281/ZENODO.15305618. (Besucht am 02. 12. 2025).

[17] Venkata Mohit Tamanampudi. “AI-Enhanced Continuous Integration and Continuous Deployment Pipelines: Leveraging Machine Learning Models for Predictive Failure Detection, Automated Rollbacks, and Adaptive Deployment Strategies in Agile Software Development”. In: 10 ().

[18] Satheesh Reddy Gopireddy. “Integrating AI into DevOps: Leveraging Machine Learning for Intelligent Automation in Azure”. In: *International Journal of Science and Research (IJSR)* 11.6 (Juni 2022), S. 2035–2039. ISSN: 23197064. DOI: 10.21275/SR22619111757. (Besucht am 03. 11. 2025).

[19] Yao Lu u. a. *Computing in the Era of Large Generative Models: From Cloud-Native to AI-Native*. Jan. 2024. DOI: 10.48550/arXiv.2401.12230. arXiv: 2401.12230 [cs]. (Besucht am 12. 11. 2025).

[20] Jeanette Uddoh u. a. “AI-Based Threat Detection Systems for Cloud Infrastructure: Architecture, Challenges, and Opportunities”. In: *Journal of Frontiers in Multidisciplinary Research* 2.2 (2021), S. 61–67. ISSN: 30509718, 30509726. DOI: 10.54660/.IJFMR.2021.2.2.61-67. (Besucht am 04. 11. 2025).

[21] Kai Petersen, Sairam Vakkalanka und Ludwik Kuzniarz. “Guidelines for Conducting Systematic Mapping Studies in Software Engineering: An Update”. In: *Information and Software Technology* 64 (Aug. 2015), S. 1–18. ISSN: 0950-5849. DOI: 10.1016/j.infsof.2015.03.007. (Besucht am 06. 11. 2025).

[22] Barbara Kitchenham und Stuart M. Charters. *(PDF) Guidelines for Performing Systematic Literature Reviews in Software Engineering*. https://www.researchgate.net/publication/30 Juli 2007. (Besucht am 06. 11. 2025).

[23] Claes Wohlin. “Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering”. In: *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*. EASE ’14. New York, NY, USA: Association for Computing Machinery, Mai 2024, S. 1–10. ISBN: 978-1-4503-2476-2. DOI: 10.1145/2601248.2601268. (Besucht am 06. 11. 2025).

[24] Karthik Puthraya, Rachit Gupta und Beverly DSouza. “The Role of Cloud-Native Architectures in Accelerating Machine Learning Workflows through Data Engineering Innovations”. In: (Feb. 2025). ISSN: 2945-3437. DOI: 10.5281/ZENODO.15106432. (Besucht am 05. 11. 2025).

[25] Abhishek Gupta und Yashovardhan Chaturvedi. “Cloud-Native ML: Architecting AI Solutions for Cloud-First Infrastructures”. In: *Nanotechnology Perceptions* 20 (Dez. 2024), S. 930–939. DOI: 10.62441/nano-ntp.v20i7.4004.

[26] Gopinath Kathiresan. “Cybersecurity Risk Modeling in CI/CD Pipelines Using Reinforcement Learning for Test Optimization”. In: *International Journal of Innovative Science and Research Technology* (Mai 2025), S. 15–25. DOI: 10.38124/ijisrt/25may339. (Besucht am 03. 11. 2025).

[27] Vijay Govindarajan. *Machine Learning Based Approach for Handling Imbalanced Data for Intrusion Detection in the Cloud Environment*. März 2025, S. 815. DOI: 10.1109/ICDT63985.2025.10986614.

[28] Rahul Amte. “Cloud-Native AI: Challenges and Innovations in Deploying Large-Scale Machine Learning Models”. In: *ISCSITR - INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING (ISCSITR-IJSRAIML) ISSN (Online): 3067-753X* 6.2 (März 2025), S. 9–18. (Besucht am 05. 11. 2025).

[29] Muhammad Talha Tahir Bajwa u. a. “CLOUD-NATIVE ARCHITECTURES FOR LARGE-SCALE AI-BASED PREDICTIVE MODELING”. In: *Journal of Emerging Technology and Digital Transformation* 4.2 (Aug. 2025), S. 207–221. ISSN: 3006-9726. (Besucht am 25. 10. 2025).

[30] Vijay Kartik Sikha. “Cloud-Native Application Development for AI- Conducive Architectures.” In: *International Journal on Recent and Innovation Trends in Computing and Communication* 11.11 (Okt. 2023).

[31] Liam Bollmann-Dodd und Álvaro Ruiz Cubero. *State of Cloud Native Development*. https://www.cncf.io/reports/state-of-cloud-native-development/. Nov. 2025. (Besucht am 07. 01. 2026).

[32] *InfoQ Cloud and DevOps Trends Report - 2025*. https://www.infoq.com/articles/cloud-devops-trends-2025/. Okt. 2025. (Besucht am 07. 01. 2026).

[33] Kirsten Nothbaum. *Cloud-native erklärt – Architektur, Vorteile und Trends 2025*. https://shop.plusserver.com/blog/was-ist-cloud-native. Sep. 2025. (Besucht am 07. 01. 2026).

[34] *Was ist CI/CD? — Automatisierung in der Softwareentwicklung*. https://www.redhat.com/de/topics/de is-ci-cd. Jan. 2025. (Besucht am 16. 12. 2025).