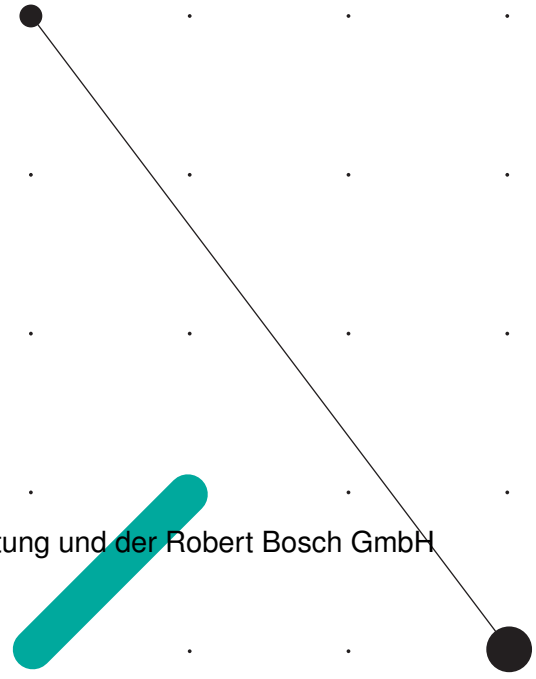
Hochschule Konstanz
Department of Computer Science

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

Cloud-Native Platform Engineering ist durch häufige Änderungen an Architektur, Betrieb und Werkzeugen sowie steigende Systemkomplexität geprägt. Der Einsatz von künstlicher Intelligenz nimmt zu, bleibt jedoch oft schwer vergleichbar und einzuordnen. Eine strukturierte Grundlage für den gezielten Einsatz solcher Verfahren fehlt bislang. Die Arbeit basiert auf einer systematischen Mapping-Studie mit definierten Such- und Auswahlkriterien. Die identifizierten Studien werden quantitativ ausgewertet und anhand zentraler Merkmale kategorisiert. Die Ergebnisse zeigen vier wiederkehrende Anwendungsfelder: Ressourcenmanagement, Plattformbetrieb, Build- und Bereitstellungsprozesse sowie Sicherheit. Über alle Felder hinweg treten ähnliche Herausforderungen auf, insbesondere Datenqualität, Integration, Governance und Kosten. Auf Basis der Ergebnisse wird ein Bewertungskonzept entwickelt, um Anwendungsfälle künstlicher Intelligenz einzuordnen. Es trennt Implementierungsaufwand und operativen Mehrwert und ergänzt praxisrelevante Zusatzdimensionen. Das Konzept wird auf der Bosch Digital Manufacturing Platform angewendet, um ein Einsatzszenario zu bewerten.

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 rund um Cloud-Native Plattformen adressiert. Diese Entwicklungen verlaufen in hohen Innovationszyklen und führen zu stetig neuen Anforderungen an Architektur und Betrieb.
Künstliche Intelligenz (KI)-gestützte Werkzeuge werden zunehmend zur Unterstützung betrieblicher Plattformprozesse eingesetzt. 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 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 betrieben werden, verglichen mit rund 30 % im Jahr 2021 [1].
Vor dem Hintergrund dieses Trends 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 [2].
Zusätzlich nimmt der Einsatz KI-gestützter Werkzeuge im Plattformkontext zu. Diese Ansätze unterscheiden sich jedoch stark im Reifegrad und erfordern je nach Lösung unterschiedliche Integrations- und Betriebsaufwände. Fachanalysen zeigen, dass die zunehmende Vielfalt an Werkzeugen Entscheidungen im Plattformbetrieb deutlich erschwert [3].
In der Praxis fehlt im Plattformbetrieb eine belastbare Grundlage, um relevante Technologieoptionen systematisch zu vergleichen. Eine Evaluierung über viele Prototypen ist meist nicht realistisch. Dadurch entsteht Unsicherheit und Ineffizienz 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 typische Anwendungsfelder sowie passende Lösungsansätze voneinander abgegrenzt.
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 Platform (BMLP) geprüft. Ergebnis der Arbeit ist eine nachvollziehbare Einordnung relevanter KI-Anwendungsfelder sowie eine daraus abgeleitete Priorisierung als Grundlage für weitere Entscheidungen im Plattformkontext.

1.3. Aufbau der Arbeit

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

tionellen Matching-Framework zusammengeführt. Kapitel 5 überträgt dieses Framework auf die BMLP und wendet es auf identifizierte Problemfelder an.
Abschließend werden die Ergebnisse in Kapitel 6 diskutiert. Kapitel 7 fasst die Arbeit zusammen und gibt einen Ausblick auf weiterführende Fragestellungen.

2

Grundlagen und verwandte Arbeiten

In diesem Kapitel werden die theoretischen Grundlagen dargestellt, die für das Verständnis der Arbeit notwendig sind. Zentrale Begriffe aus dem Bereich Cloud-Native Platform Engineering, DevOps, CI/CD, den Lernparadigmen des maschinellen Lernens sowie AIOps werden in Abschnitt 2.1 erläutert. Anschließend werden in Abschnitt 2.2 relevante verwandte Arbeiten beschrieben und kritisch eingeordnet.

2.1. Grundlagen

Dieser Abschnitt ordnet die für die Arbeit relevanten Konzepte aus Cloud-Native Technologien, DevOps sowie KI im Plattformbetrieb ein. Sie bilden die fachliche Grundlage für die anschließende Literaturanalyse und die Auswertung der KI-Anwendungen im Platform Engineering.

2.1.1. Cloud-Native Technologien und Platform Engineering

Cloud-Native gewann ab 2013 insbesondere durch die Etablierung von Containertechnologien bis hin zu Kubernetes an Bedeutung. Im Kern beschreibt Cloud-Native heute weniger eine einzelne Technologie, sondern ein Zielbild für modular aufgebaute Systeme (z.B. Mikroservices), die sich gut bereitstellen, skalieren und resilient betreiben lassen. Die Cloud Native Computing Foundation (CNCF) (Stiftung für Cloud-Native-Computing) fasst Cloud-Native als Ansatz für skalierbare Anwendungen in dynamischen Cloud-Umgebungen zusammen [4].

Kubernetes hat sich dabei als zentrale Plattform etabliert. Es dient als portable, erweiterbare 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 [5, 6].
Im Platform Engineering werden diese Grundlagen genutzt, um Entwicklerteams über eine interne Plattform (Internal Developer Platform) standardisierte, sichere Selbstbedienungsfunktionen (Self-Service-Capabilities) bereitzustellen. Damit soll die Lieferfähigkeit, Compliance und Betrieb verbessert werden [7].

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 [8, 9].
Continuous Integration (CI) und Continuous Delivery (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 [10, 11].

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 [12, 13].
UL arbeitet mit ungelabelten Daten und zielt darauf ab, Strukturen, Cluster oder Zu-

sammenhänge zu erkennen, ohne dass eine explizite Zielvariable vorgegeben ist [12, 14].
RL beschreibt Verfahren, bei denen ein Agent durch Interaktion mit einer Umgebung Handlungen lernt, um eine langfristige (kumulative) Belohnung zu maximieren [12, 15]. 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
DL / 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.

2.1.4. AIOps und verwandte Konzepte

Artificial Intelligence for IT Operations (AIOps) bezeichnet den Einsatz von KI-Technologien zur Automatisierung und Optimierung von IT-Betriebsprozessen [16, 17]. Im Kontext dieser Arbeit liegt der Fokus auf der Anwendung von KI zur Überwachung, Analyse und Optimierung von Cloud-Native Plattformen. 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 heterogener Betriebsdaten. Zur Einordnung wird AIOps von Machine Learning Operations (MLOps)[1] abgegrenzt.

[1] MLOps bezeichnet Ansätze und Praktiken zur Entwicklung, zum Training, zur Bereitstellung und zum Betrieb von Modellen des maschinellen Lernens und adressiert primär den Lebenszyklus von KI-Modellen.

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 [16, 18]. Weitere Arbeiten adressieren prädiktive Skalierung, um Lastverläufe vorherzusagen und Ressourcen vorausschauend anzupassen [19]. Auch RL wird zur dynamischen Steuerung der Lastverteilung und zur Erhöhung der Fehlertoleranz eingesetzt [20].
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 [12, 21, 22]. Ergänzend wird AIOps als Ansatz beschrieben, um komplexe Cloud-Infrastrukturen durch automatisierte Analyse und Monitoring besser beherrschbar zu machen [17, 23]. Sicherheitsbezogene Aspekte werden von Uddoh u. a. [24] 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 [4].
Kritisch ist festzuhalten, dass viele Arbeiten einzelne Anwendungsfälle isoliert betrachten oder stark werkzeug- bzw. monitoring-getrieben sind. Zudem existieren thematisch nahe Beiträge mit MLOps- oder Datenpipeline-Fokus, die den Plattformbetrieb aus Sicht von Platform Engineers nur indirekt adressieren und daher ausgeklammert wurden. Eine systematische, übergreifende Einordnung entlang typischer Aufgaben im Platform Engineering sowie eine Bewertung von Übertragbarkeit und praktischem Nutzen bleibt häufig offen.

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 werden die spezifischen Ziele und Forschungsfragen dieser Arbeit definiert. Abschnitt 3.2 beschreibt den Prozess der Literaturanalyse, einschließlich Suchstrategie, Ein- und Ausschlusskriterien, Schneeballmethode sowie Datenerhebung und -analyse. In Abschnitt 3.3 wird das Vorgehen der Mapping-Studie beschrieben und die Auswertung der identifizierten Studien erläutert.

3.1. Forschungsfragen

Die Forschungsfragen werden durch ein strukturiertes methodisches Vorgehen beantwortet. Jede Forschungsphase ist darauf ausgelegt, das Verständnis über den Einsatz von KI im Platform Engineering schrittweise zu vertiefen. Die Mapping-Studie 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-Studie 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 be-

stehen 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-Plattform-Umgebungen übertragen und hinsichtlich ihres Mehrwerts und ihrer Umsetzbarkeit bewerten?
Ziel: Entwicklung eines Bewertungsschemas, das die Passung zwischen KI-Lösungen und spezifischen Plattform-Use-Cases beschreibt und die praktische Umsetzbarkeit aufzeigt.

3.2. Literaturanalyse-Prozess

Der folgende Abschnitt beschreibt den Ablauf der Literaturanalyse im Rahmen der durchgeführten Mapping-Studie. 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.

3.2.1. Suchstrategie

Zur Durchführung der Mapping-Studie 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”, “AIOps”, “Artificial Intelligence”, “Machine Learning”, “DevOps” und “Kubernetes Cluster”.

Zur Transparenz und Vollständigkeit sind die exakten Suchstrings sowie ihre logischen Verknüpfungen im Anhang dokumentiert.

3.2.2. 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 Whitepaper, 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).

3.2.3. Schneeballmethode

Zur Ergänzung der systematischen Suche wurde eine Schneeballmethode nach den Leitlinien von Wohlin [25] 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.

3.2.4. 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.

3.2.5. 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-Studie dargestellt. Diese Übersicht zeigt, in welchen Themenfelder bereits Forschungsschwerpunkte existieren und wo noch Forschungslücken bestehen.

3.3. Vorgehen der Mapping-Studie

Für diese Arbeit wird eine Mapping-Studie nach den Richtlinien von Petersen u. a. [26] 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 [27] berücksichtigt. In der Durchführungsphase werden identifizierte Studien anhand festgelegter Ein- und Ausschlusskriterien geprüft. Zusätzlich wird das Schneeballverfahren nach Wohlin [25] 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 Analyse umfasst dabei Mappings zwischen Anwendungsfeldern und Herausforderungen, zwischen Lernparadigmen und verwendeten Algorithmen sowie zwischen Anwendungsfeldern und genutzten Datenquellen.
Die Ergebnisse der Mapping-Studie bilden die Grundlage für die anschließende Analyse der BMLP sowie für die Entwicklung eines Frameworks zur Bewertung von KI-Potenzialen in Cloud-Native Plattformumgebungen.

4

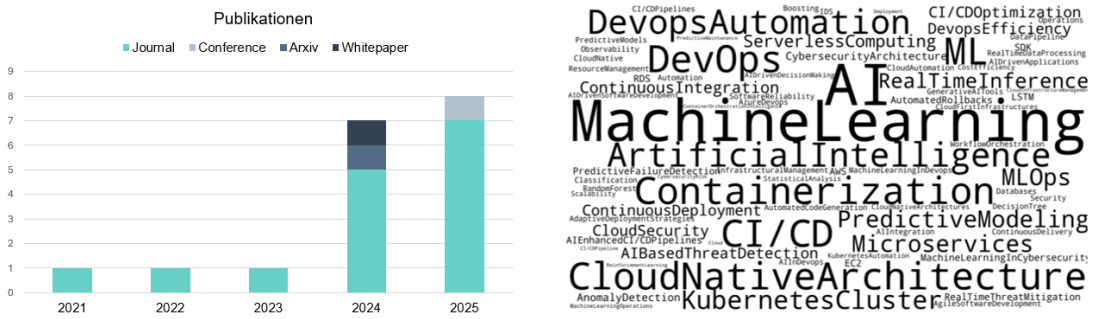
Ergebnisse der Literaturanalyse

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 Mapping-Studie 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.

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. Der zeitliche Anstieg der Publikationen fällt mit den jüngsten Fortschritten im Bereich generativer KI zusammen [23].
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.



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

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. Maßnahmen in diesem Bereich haben häufig einen direkten, messbaren Einfluss auf Kosten und Performance, etwa durch eine bessere Ressourcenauslastung oder stabilere Laufzeiten [28]. 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 [22]. Die Optimierung von CI/CD-Pipelines und Sicherheits- und Bedrohungserkennung sind mit jeweils drei Publikationen vertreten. Beide Anwendungsfelder werden in vielen Arbeiten aufgegriffen, stehen jedoch seltener im Fokus der jeweiligen Studie. Die Optimierung der CI/CD-Pipeline wird oft als Teilaspekt in ein größeres Gesamtbild der Infrastruktur-Transformation behandelt [17].
Abbildung 4.2 zeigt diese Verteilung. Zu beachten ist, dass viele Studien mehr als einen Bereich ansprechen. In der Praxis überschneiden sich die Anwendungsbereiche häufig, da KI-Lösungen oft mehrere Aufgaben gleichzeitig unterstützen. Für die Vergleichbarkeit wurde jedoch jeweils der dominante Use Case pro Publikation ausgewählt.

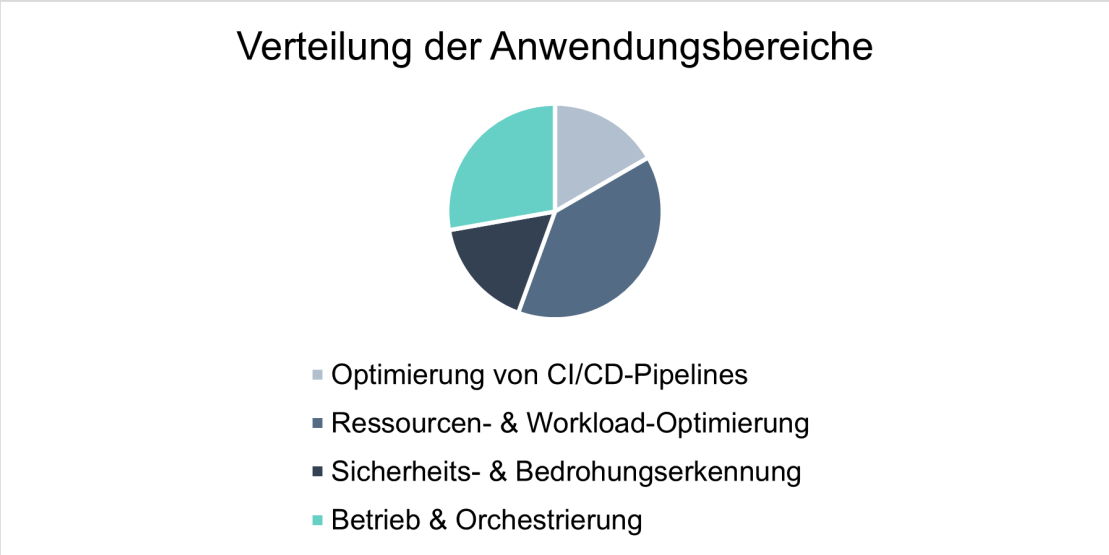


Abbildung 4.2: Verteilung der Anwendungsbereiche

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 Verteilung 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.
Die Ergebnisse zeigen ein Spannungsfeld. KI wird eingesetzt, um Plattformen effizienter und kostengünstiger zu betreiben. Gleichzeitig verursachen Training, Inferenz sowie zusätzliche Überwachungs- und Datenpipelines selbst spürbare Ressourcen- und Betriebskosten [4]. 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. Vielmehr wird er stark durch Anforderungen an Datenzugriff, Nachvollziehbarkeit und Regelkonformität begrenzt [21].

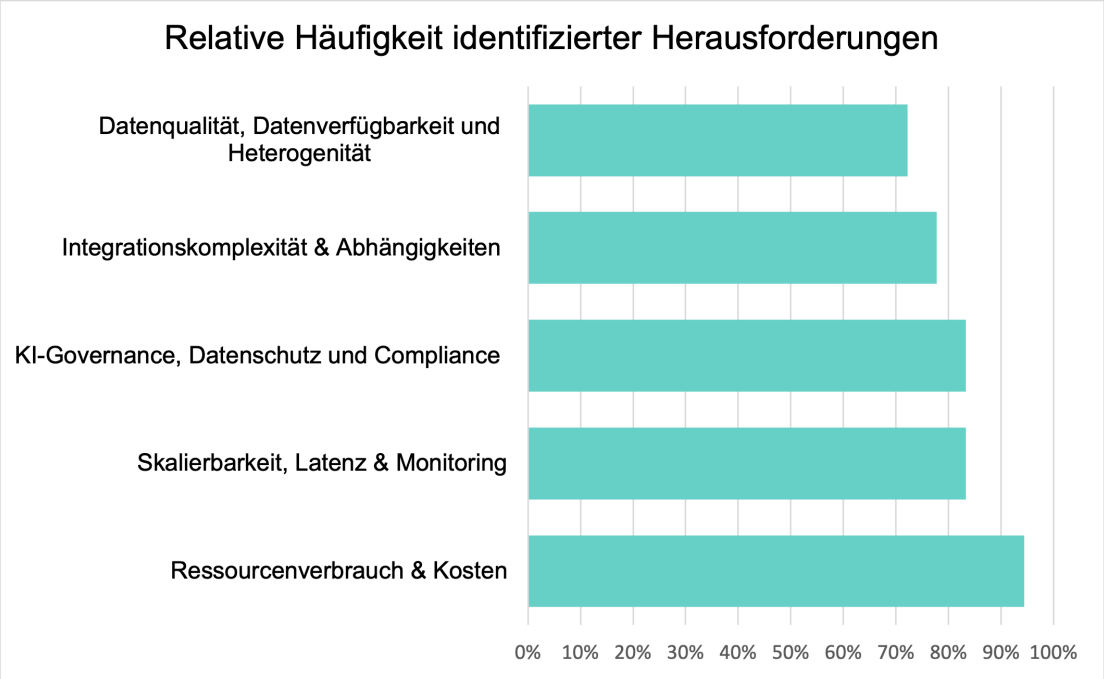


Abbildung 4.3: Relative Häufigkeit der Herausforderungen

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.
Die Auswertung zeigt, dass SL in den meisten Arbeiten eine zentrale Rolle spielt. Es wird in vier Publikationen ausdrücklich beschrieben, elfmal implizit erkennbar und in zwei kurz erwähnt. RL wird insgesamt siebenmal explizit genannt und in vier weiteren Arbeiten erwähnt. UL tritt mit drei expliziten und sieben impliziten Nennungen seltener auf, ist jedoch ebenfalls präsent.
Die Verteilung ist in der folgenden Abbildung 4.4 dargestellt.

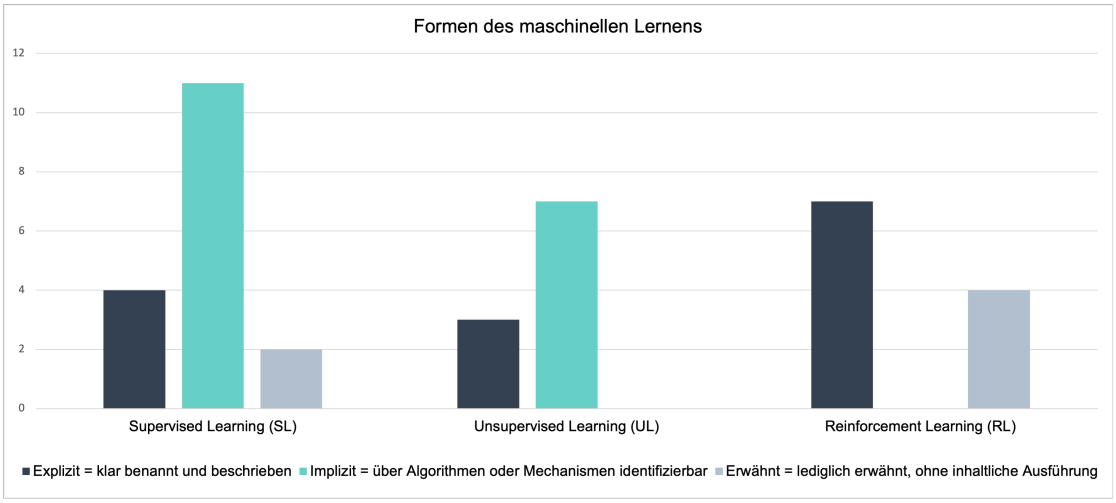


Abbildung 4.4: Formen des maschinellen Lernens

Auffällig ist die deutliche Differenz zwischen explizit und implizit erkennbaren Anwendungen, insbesondere bei SL und UL. Dies zeigt, dass viele Publikationen die entsprechenden Paradigmen beschreiben, ohne die zugrunde liegende Lernform ausdrücklich zu benennen. Ein Erklärungsansatz für die Dominanz von SL 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 [21].

4.1.4. Verwendete Algorithmen

Die in den Publikationen verwendeten Algorithmen lassen sich den jeweiligen Lernparadigmen zuordnen. Die Analyse zeigt, dass im Kontext des Platform Engineerings eine Vielfalt an KI- und Verfahren des maschinellen Lernens (ML) eingesetzt wird. Für eine systematische Bewertung wurde eine eigene Kategorisierung entwickelt. Den Ausgangspunkt bildet die Tabelle aus Enemosah [12], 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 DL / NN, 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 NN die dominierende Methodenklasse bilden. In der Literatur dominieren DL-Verfahren, insbesondere aufgrund ihrer Eignung zur Verarbeitung heterogener Daten wie Logs, Metriken und Text. Gleichzeitig zeigen die Arbeiten, dass für einzelne Aufgaben auch einfachere Modelle ausreichend sind, diese jedoch seltener im Fokus der Forschung stehen [12, 21].
Diese Verteilung ist in der Abbildung 4.5 dargestellt.

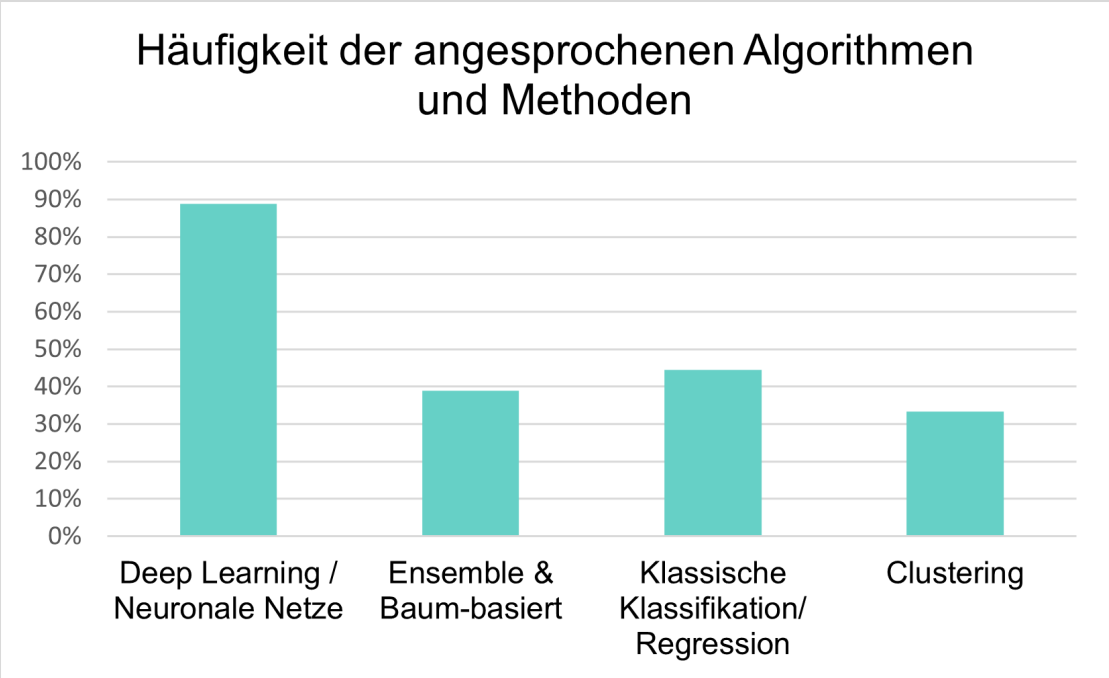


Abbildung 4.5: Verteilung der Algorithmen und angesprochenen Methoden

4.2. Ergebnisse der Mapping-Studie

In diesem Abschnitt werden die Ergebnisse vertieft und mit einer Kreuztabellenanalyse präsentiert. Diese Zuordnungen untersuchen Beziehungen zwischen Anwendungsfeldern und Herausforderungen (4.2.1), Lernparadigmen und Algorithmen (4.2.2) sowie Anwendungsfelder und Datenquellen (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.
Besonders deutlich wird die starke Verknüpfung der Use Cases Ressourcen- 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 [19]. Echtzeit-Inferenz verkürzt die Reaktionszeit. Gleichzeitig steigt durch höheren Verbrauch von Central Processing Unit (CPU) und Speicher der Ressourcen- und Kostenaufwand [29].
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 Ressourcenverbrauch 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 Optimierung der CI/CD-Pipeline 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 [21].
Insgesamt verdeutlicht die Verteilung der Häufigkeiten, dass KI-Anwendungen im Plattform 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.

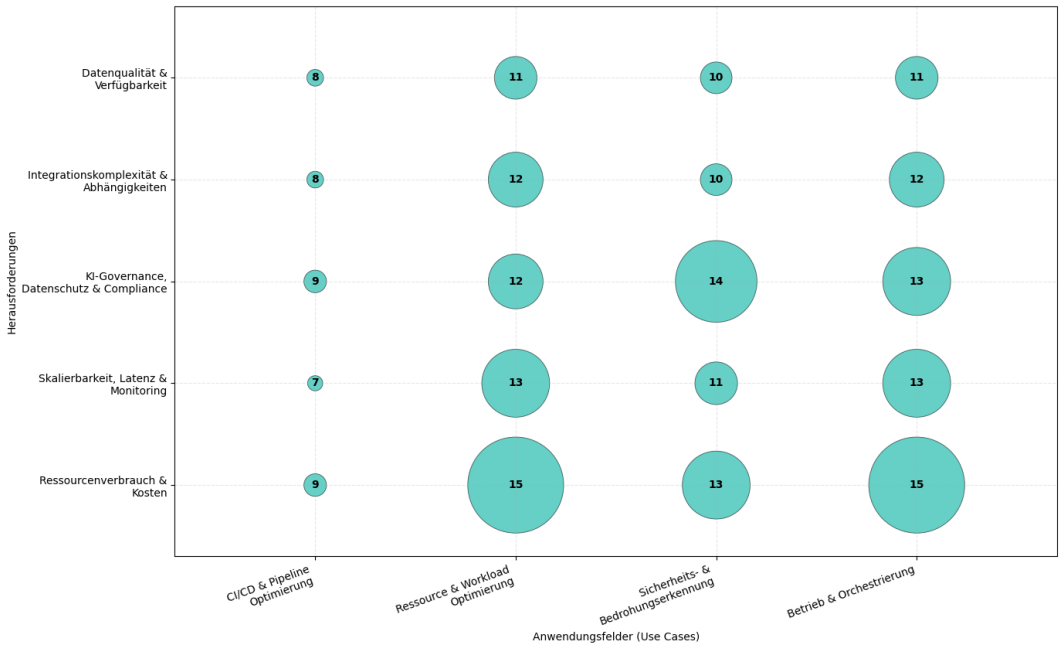


Abbildung 4.6: Korrelation zwischen Anwendungsfeldern und Herausforderungen

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.

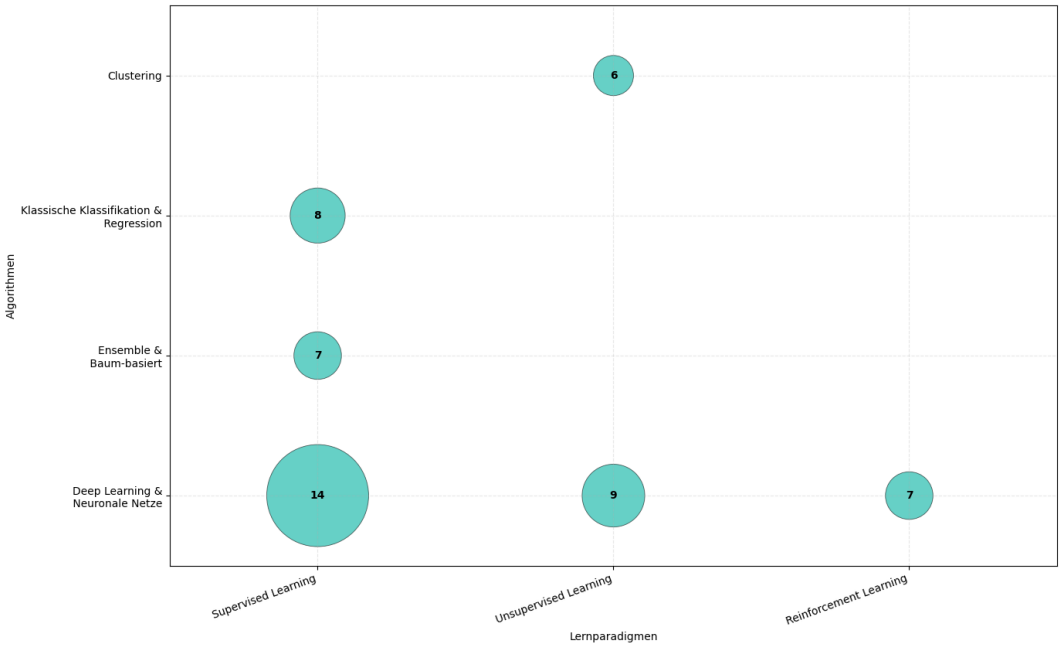


Abbildung 4.7: Korrelation zwischen Lernparadigmen und Algorithmen

SL weist im Mapping die größte Bandbreite auf. Es wird am häufigsten mit DL / NN kombiniert ($n = 14$), gefolgt von Klassifikation / Regression ($n = 8$) sowie Ensemble- und baum-basierten Verfahren ($n = 7$). Damit dominiert SL sowohl in der Anzahl der Nennungen als auch in der Vielfalt der berichteten Verfahren. Mehrere Arbeiten beschreiben SL als zentralen Ansatz zur Optimierung einzelner Pipeline-Schritte, etwa für Test-, Build- oder Deploy-Entscheidungen [12, 21].
UL zeigt hingegen ein engeres Spektrum. Im Mapping dominieren DL-basierte Ansätze ($n = 9$) sowie Clustering ($n = 6$). Der Schwerpunkt liegt dabei auf der Muster- und Anomalieerkennung. Mehrere Arbeiten zeigen, dass sich das erlernte Normalverhalten in dynamischen Plattformumgebungen häufig verändert und ohne Anpassung der Modelle vermehrt Fehlalarme entstehen [12, 21].
RL tritt in mehreren Studien auf, wird jedoch selten algorithmisch konkretisiert. Berichtet werden ausschließlich Kombinationen mit DL / NN ($n = 7$). Zwar werden in einzelnen Arbeiten konkrete RL-Algorithmen genannt, zugleich beschreiben die Studien jedoch einen

hohen Rechen- und Zeitaufwand sowie eine komplexe Implementierung als zentrale Herausforderungen [12, 20]. Insgesamt bleibt die Zuordnung zu spezifischen Algorithmen 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.

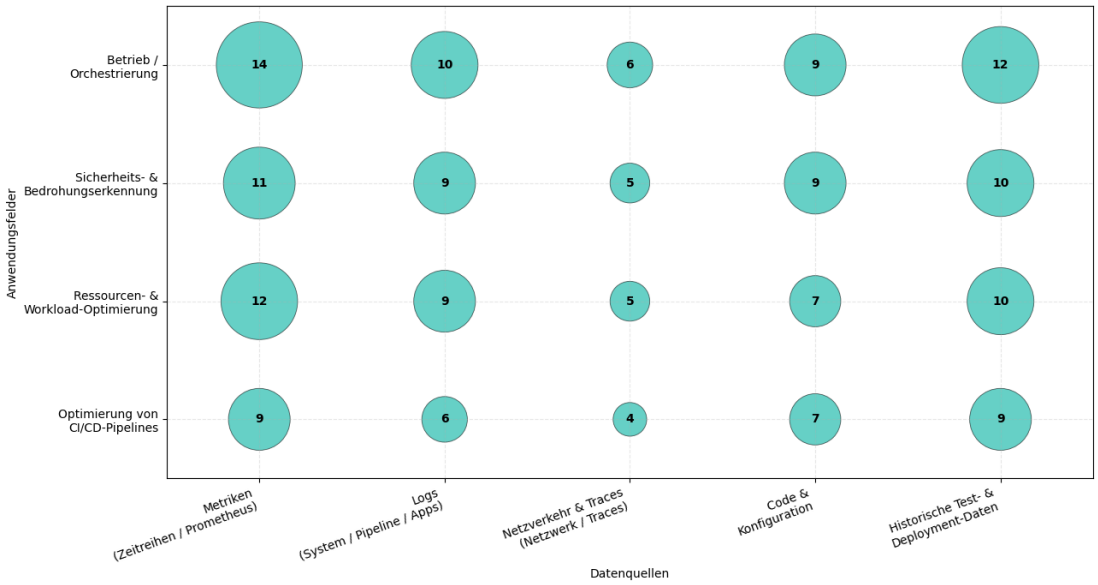


Abbildung 4.8: Korrelation zwischen Anwendungsfeldern und Datenquellen

Metriken beschreiben den quantitativen Systemzustand, etwa CPU, Speicher oder Latenzen. Sie stammen häufig aus Monitoring-Systemen wie Prometheus[1] oder CloudWatch[2]. Entsprechend treten sie in nahezu allen Use Cases stark auf. Besonders ausgeprägt sind sie in Betrieb / Orchestrierung ($n = 14$) sowie in der Ressourcen- und Workload-Optimierung ($n = 12$) [16, 19].
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 ($n = 6$ bis 10), mit hohen Werten in Betrieb / Orchestrierung ($n = 10$) und sicherheitsnahen Szenarien ($n = 9$). Betriebsstörungen, Fehlkonfigurationen oder

[1] https://prometheus.io/docs/introduction/overview/
[2] https://aws.amazon.com/de/cloudwatch/

sicherheitsrelevante Ereignisse lassen sich in Logdaten erkennen [30].
Netzwerkdaten und Traces ergänzen diese Sicht um Kommunikationsbeziehungen und Laufzeitpfade verteilter Systeme (z. B. Flow-Daten oder OpenTelemetry-Traces[3]). 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 Mikroservice-Architekturen) oder wenn Anomalien über Kommunikationsmuster erkannt werden sollen [4].
Code und Konfiguration (Quellcode, Container-Artefakte, Infrastructure-as-Code-Definitionen) werden wichtig, wenn KI nicht nur Symptome bewertet, sondern Änderungen und Konfigurationsstände einbezieht. Im Mapping zeigt sich eine breite Nutzung über alle Use Cases ($n = 7$ bis 9). Auffällig ist die hohe Ausprägung bei Betrieb / Orchestrierung und Sicherheit (je $n = 9$), was zu riskanten Änderungen, unsicheren Voreinstellungen oder Fehlkonfigurationen passt [17].
Historische Test- und Deployment-Daten (Build-Historien, Testergebnisse, Deployment-Verläufe) treten vor allem dort auf, wo Verfahren aus vergangenen Ausführungen lernen. Im Mapping zeigen sich erhöhte Werte bei Betrieb / Orchestrierung ($n = 12$), Ressourcenoptimierung ($n = 10$) und CI/CD-Optimierung ($n = 9$) [21].
Deutlich wird, dass Betrieb / Orchestrierung die stärksten Überschneidungen mit nahezu allen Datenquellen aufweist, weil hier Monitoring-Daten und Änderungsdaten zusammenlaufen. Ohne dieses Gesamtbild gelingt eine saubere Steuerung KI-gestützter Abläufe nur eingeschränkt. Das erhöht die Wahrscheinlichkeit von Fehlalarmen und erschwert die Eingrenzung der eigentlichen Ursache.

[3] https://opentelemetry.io

4.3. Matching-Framework

Aufbauend auf den quantitativen Ergebnissen aus Abschnitt 4.1 sowie den Mustern der Mapping-Studie in Abschnitt 4.2 wird in diesem Abschnitt ein konzeptionelles Matching-Framework abgeleitet. Ziel ist es, die identifizierten KI-Ansätze zu übertragbaren Anwendungsmustern für generische Use Cases zu überführen.

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**
Konkrete Plattform-aufgabe	Automatisierte Anpassung von Instanztypen (EC2/RDS) oder Kubernetes-Pods (HPA) basierend auf der Vorhersage von CPU-/RAM-Spitzen.
Empfohlene KI-Methode	DL (z.B. LSTM) zur Zeitreihenprognose oder RL zur dynamischen Lastverteilung in Echtzeit.
KPI/Mehrwert	Ressourcenauslastung bis zu +25 % (RL vs. Heuristik) [19], Kosteneffizienz durch dynamische Allokation mit bis zu 55 % geringeren Cloud-Kosten [28].
Limitationen	Hoher Rechenaufwand für das Modelltraining (z.B. LSTMs), Startverzögerungen bei initialer Bereitstellung (Serverless), benötigt ca. 2 Monate historische Daten [19, 31].
Einsatzvoraussetzungen	Mind. 2 Monate historische Zeitreihen-Telemetrie aus CloudWatch/Prometheus sowie automatisierter API-Schreibzugriff für Instanz- bzw. Skalierungsanpassungen [19].
Konkrete Tools	AWS CloudWatch, Boto3 SDK, Kubernetes HPA, KEDA (Event-Driven Autoscaling), Knative, Megalix [4, 17, 19, 32].

In der Literatur werden dafür vor allem Verfahren zur Auswertung zeitlicher Verläufe sowie lernbasierte Ansätze zur Entscheidungsunterstützung beschrieben [19]. 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 [28]. 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.
Es wird zwischen Betrieb auf eigener Infrastruktur (On-Premises) und Betrieb in externen Cloud-Umgebungen (Public Cloud) unterschieden. Die praktische Relevanz des Musters hängt vom Betriebsmodell ab. In der Public Cloud fängt automatische Skalierung viele Lastspitzen bereits ab, da zusätzliche Kapazität kurzfristig bereitgestellt werden kann. Im On-Premises-Betrieb ist der Ansatz häufig relevanter. Feste Kapazitätsgrenzen und längere Beschaffungszeiten lassen Fehlentscheidungen bei der Ressourcendimensionierung stärker wirken.

4.3.2. Automatisierte Release-Absicherung

Das Muster Automatisierte Release-Absicherung adressiert die Absicherung von Rollouts, 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 des Rollouts 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**
Konkrete Plattformaufgabe	Automatisiertes Rollback bei Canary-Deployments (schrittweiser Rollout) durch Überwachung von Abweichungen in Latenz und Fehlerraten (Release-Gating).
Empfohlene KI-Methode	UL (Autoencoder) zur Anomalieerkennung oder RL (z.B. DQN) zur Optimierung des Rollback-Zeitpunkts [21].
KPI/Mehrwert	Mean Time To Recovery (MTTR) bis zu -40 %; Systemverfügbarkeit bis zu +18 %; Vorfallerkennung um bis zu 35 % beschleunigt [16, 20, 21].
Limitationen	Risiko von False Positives (unnötige Rollbacks); zusätzlicher Validierungs- und Governance-Aufwand bei RL-Entscheidungen (z.B. Nachvollziehbarkeit und Erklärbarkeit).
Einsatzvoraussetzungen	Etablierte Baseline für Normalverhalten in Service-Mesh-gestützten Telemetriedaten (Traffic-Steuerung & Beobachtbarkeit; Latenz, Fehlerraten) sowie automatisierte Rollback-Mechanismen und Echtzeit-Streaming der Deployment-Metriken [21, 32].
Konkrete Tools	Spinnaker / Spinnaker AI, Harness, StackStorm, Dynatrace, LaunchDarkly, Sentry, Istio, Linkerd [17, 32, 33].

Verfahren zur Erkennung von Abweichungen vom normalen Systemverhalten kommen zum Einsatz, um problematische Änderungen frühzeitig zu identifizieren [21]. 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 [16, 20, 21]. 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 eine stabile Referenz für das Normalverhalten sowie automatisierte Abbruch- und Wiederherstellungsmechanismen und eine kontinuierliche Erfassung relevanter Rollout-Metriken 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**
Konkrete Plattformaufgabe	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	SL (z.B. Random Forest, XGBoost) zur Klassifikation von Fehlerrisiken und Fehlertypen auf Basis historischer Build-Metadaten und Log-Features [16, 21].
KPI/Mehrwert	Deployment-Zeit bis zu -30 %; Vorhersagegenauigkeit ca. 87 %; Reduzierung der Fehlersuchzeit bis zu 40 % [16, 21].
Limitationen	Geringe Anzahl von Fehlerfällen (unausgewogene Trainingsdaten), Konzeptdrift bei Änderungen an Build-Pipelines oder Abhängigkeiten, Regelmäßige Überwachung und erneutes Training erforderlich
Einsatzvoraussetzungen	Zentral verfügbare und einheitlich strukturierte Build- und Test-Logs, ausreichend große, gelabelte Historie (Erfolg vs. Fehlertyp). Eindeutige Referenz auf Commit, Pipeline-Schritt und Artefaktversion [21, 30].
Konkrete Tools	K8sGPT, Jenkins X (mit AI-Plugins), CircleCI Insights, DeepCode, SonarQube, GitLab CI, GitHub Copilot [4, 12, 16, 17, 33].

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 [16, 21, 34]. Ein zentrales Ziel ist dabei die Reduktion der mittleren Wiederherstellungsdauer (MTTR) nach fehlgeschlagenen Builds. 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 [16, 21]. 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.
Als methodischer Ansatz werden in der Literatur sowohl DL-Modelle (z.B. CNNs) als auch Gradient-Boosting-Verfahren (z.B. XGBoost) eingesetzt. XGBoost kann dabei besonders dann stabil funktionieren, wenn es nur wenige echte Sicherheitsvorfälle in den Daten gibt [24, 34]. 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 [24]. Gleichzeitig kann der Betrieb aufwendig werden, weil DL-Modelle beim Einsatz zusätzliche Rechenleistung benötigen und damit Kosten verursachen. Zusätzlich steigen 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.

Tabelle 4.4: Übersicht zum Muster DevSecOps (Security Ops)

DevSecOps (Security Ops)	**Beschreibung**
Konkrete Plattformaufgabe	Priorisierung von Sicherheits-Scans sowie Echtzeit-Erkennung von Angriffen (z.B. DDoS, Malware) anhand von Netzwerk- und Audit-Telemetrie.
Empfohlene KI-Methode	DL (z.B. CNNs auf flow-basierten Merkmalsrepräsentationen) oder Gradient Boosting (z.B. XGBoost) für unausgewogene Sicherheitsdaten [24].
KPI/Mehrwert	Erkennungsgenauigkeit bis ca. 95 %; Reduktion von Fehlalarmen um ca. 20 %; hohe Detektionsraten auch für neuartige Angriffsmuster (studienabhängig) [24].
Limitationen	Hoher Rechenbedarf (insbesondere bei DL-Inferenz); Datenschutz- und Compliance-Risiken (DSGVO) bei der Analyse paketbasierter Protokolle und potenziell personenbezogener Daten.
Einsatzvoraussetzungen	Echtzeit-Erfassung von Netzwerkverkehr (z.B. Flow-Daten) und Protokolldaten. Externe Bedrohungsinformationen zur Kontextualisierung und Priorisierung [24, 30].
Konkrete Tools	IBM Watson, Microsoft Sentinel, Amazon GuardDuty, Trivy, Grype, Snyk, Falco, Aqua Security, Kyverno, Clair [4, 16, 33]

5

Anwendung der Literaturrecherche

Abschnitt 5.1 beschreibt das zugrunde liegende Bewertungskonzept. Abschnitt 5.2 ordnet dieses in den Kontext der BMLP ein. In Abschnitt 5.3 wird das Konzept exemplarisch angewendet. Daraus wird in Abschnitt 5.4 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. Aspekte 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 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 typischerweise Vorarbeiten, etwa bei Daten- und Integrationsgrundlagen. Anwendungsfälle mit hohem Aufwand und geringem Mehrwert sollten nur weiterverfolgt werden, wenn klare Abhängigkeiten bestehen.

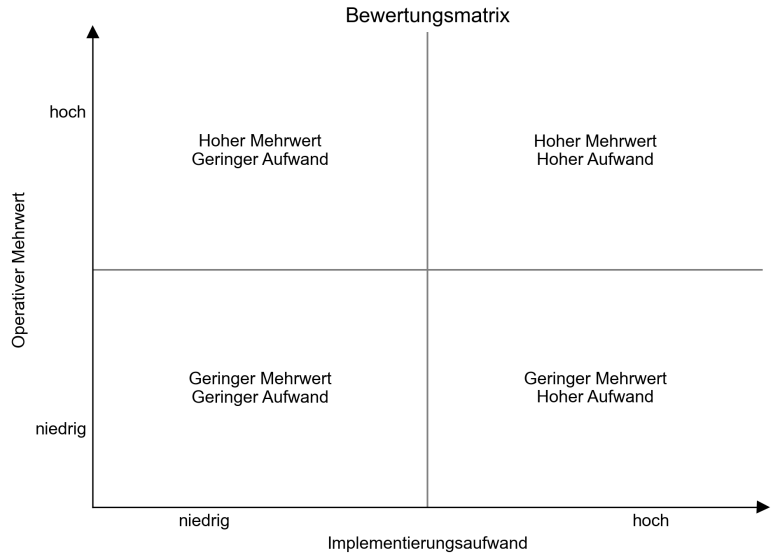


Abbildung 5.1: Bewertungsmatrix entlang von Implementierungsaufwand und operativem Mehrwert

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, 16, 22]
	Technischer Fit	Lässt sich der Use Case realistisch in bestehende Plattform-, oder CI/CD-Architekturen integrieren?	X	2	[18, 19]
	Erforderliche Fachkenntnisse	Ist ausreichende Expertise für Entwicklung, Betrieb und Wartung KI-gestützter Funktionen vorhanden?	X	1	[12, 22]
Betriebswirksamkeit & Skalierbarkeit	Stabilität und operativer Nutzen	Erzielt die Lösung im laufenden Plattformbetrieb einen stabilen, messbaren Mehrwert?	Y	2	[18, 19, 20]
	Skalierbarkeit und Wiederverwendbarkeit	Lässt sich der Ansatz ohne hohen Individualaufwand auf weitere Services oder Teams übertragen?	Y	1	[12, 16]
Compliance & Governance	Datenschutz und Datenzugriff	Welche Anforderungen ergeben sich aus Datenschutz, Datenklassifikation und Zugriffskontrolle?	X	2	[16, 24]
	Organisatorische Verankerung	Sind Verantwortlichkeiten, Freigaben und Governance-Strukturen klar definiert und akzeptiert?	X	1	[12, 17]
Reifegrad	Technologiereife	In welchem Stadium befindet sich die Lösung (Konzept, Pilot, Produktivbetrieb)?	X+Y	1	[17, 34]
	Betriebsreife	Ist der dauerhafte Betrieb durch klare Zuständigkeiten, Überwachungsmechanismen und eine ausreichende Betriebsdokumentation abgesichert?	X+Y	1	[16, 18, 19]

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. Die Berechnung erfolgt 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 Python-Programm 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 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. Der Betrieb umfasst Edge-Standorte in den Werken, zentrale Rechenzentren sowie cloudbasierte Instanzen innerhalb der Bosch-internen Cloud-Infrastruktur. Die Plattform ist auf einen hybriden Betrieb ausgelegt und adressiert Anforderungen an Latenz, Verfügbarkeit und Compliance gleichermaßen.
Technisch basiert die Plattform auf Container-Orchestrierung mit Kubernetes, deklarativen Schnittstellen sowie automatisierten CI/CD-Prozessen und durchgängiger Beobachtbarkeit. 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 basieren auf internen Plattformdokumentationen und Expertengesprächen (nicht öffentlich zugänglich).

5.2.1. Architektur und Betriebsmodell

Fachliche Funktionen in der BMLP werden als eigenständige Module in einer verteilten und modularen Grundstruktur umgesetzt. 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.
In jedem Werk wird ein eigenständiges Kubernetes-Cluster betrieben. Ergänzend existieren zentrale Cluster für übergreifende Plattformdienste sowie separate Werks-Cluster für latenzkritische Funktionen. So werden Echtzeitanforderungen sichergestellt und zentrale Cluster im Multi-Tenant-Betrieb genutzt.
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 werden plattformweit einheitlich umgesetzt.

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, Konfigurationsverwaltung sowie die Verwaltung von Schlüsseln und Zugangsdaten zu einem durchgängigen Prozess. Die Bereitstellung erfolgt reproduzierbar und nachvollziehbar unter Berücksichtigung regulatorischer Anforderungen.
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 Referenzen 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 BMLP 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 BMLP 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. Es liegen keine konsolidierte Kennzahlen vor. Aus Betriebserfahrungen ergibt sich, dass diese Abbrüche wiederkehrend auftreten. Die Auswirkungen sind insbesondere in Umgebungen mit paralleler Entwicklung und hoher Änderungsfrequenz sichtbar. Fehlerhafte Builds blockieren nachgelagerte Schritte und machen erneute Pipeline-Läufe erforderlich. Dadurch verschiebt sich die Bereitstellung von Funktionen und beeinträchtigt die Änderungsbereitstellung der Plattform insgesamt.

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. Dazu zählen 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 zwischen einzelnen Komponenten und Schritten. Beispiele hierfür sind 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 geeignete 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 durchgeführten Analyse und Anwendung des Bewertungsrasters.

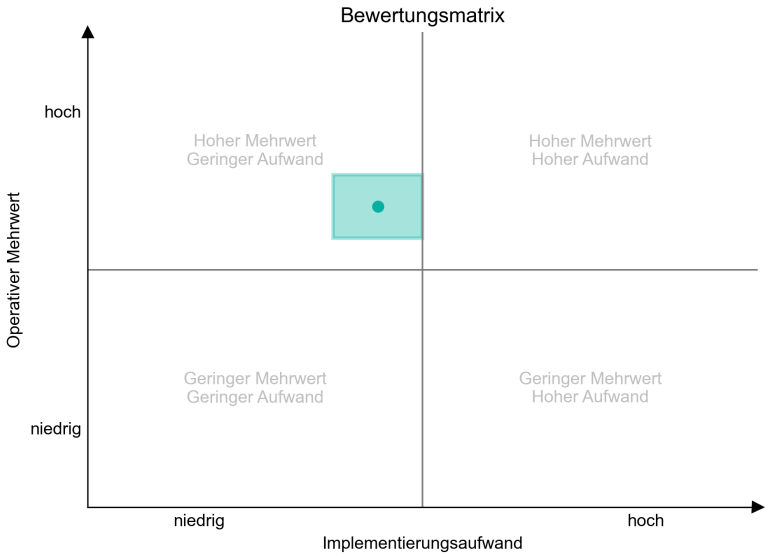


Abbildung 5.2: Einordnung des Use Case in die Bewertungsmatrix

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. Die notwendigen Datenquellen sind bereits vorhanden, und es sind keine grundlegenden Änderungen an bestehenden Abläufen erforderlich.
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 [16]. 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.

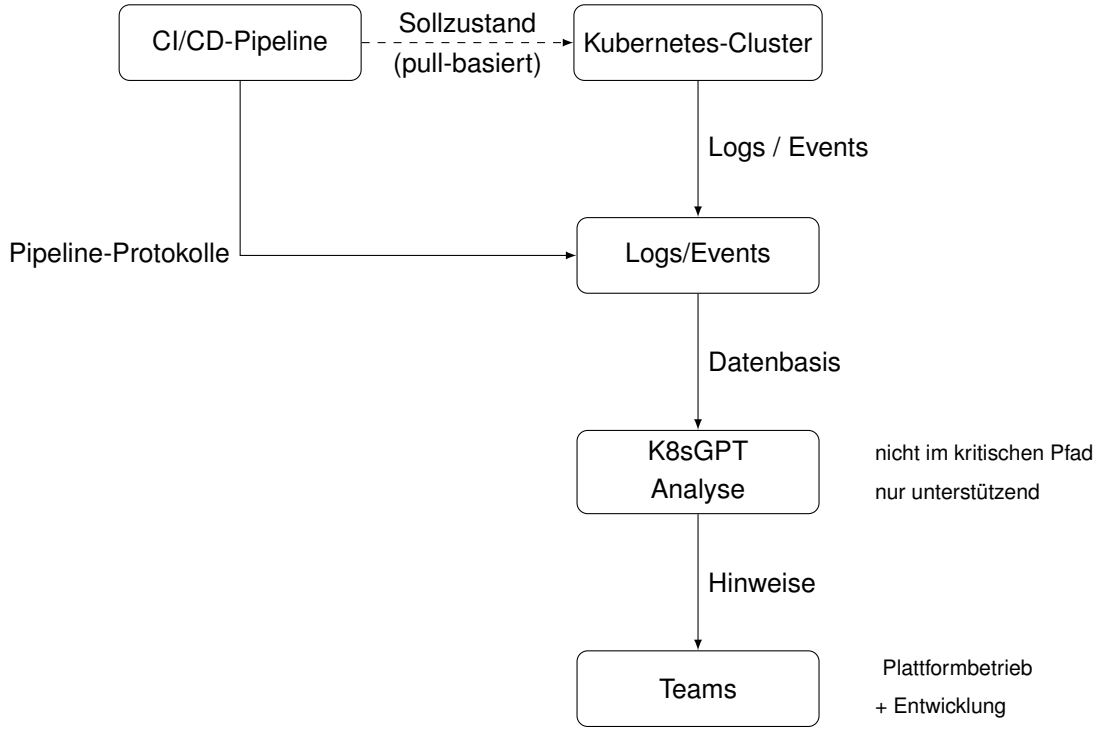


Abbildung 5.3: Einordnung von K8sGPT im Plattformbetrieb

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

In diesem Kapitel werden die Ergebnisse der Arbeit eingeordnet und in Bezug auf die Forschungsfragen zusammengeführt. Zunächst erfolgt die Beantwortung der Forschungsfragen in Kapitel 6.1. Anschließend werden die Limitationen der Arbeit in Kapitel 6.2 dargestellt.

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 Proaktive 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 [18, 19].

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 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 relevant ist. Der Ansatz wird unzuverlässig, wenn sich das Normalverhalten durch Releases und Konfigurationsänderungen ständig verschiebt und dadurch viele Fehlalarme entstehen [18, 21].

Die Absicherung von Rollouts zielt darauf ab, fehlerhafte Versionen früh zu begrenzen. Während einer schrittweisen Rollout-Phase werden Fehlerraten und Antwortzeiten beobachtet und mit dem erwarteten Normalzustand verglichen. Bei auffälligen Abweichungen kann ein Rollout gestoppt oder auf die vorige Version zurückgesetzt werden. Das ist relevant, weil Rollouts regelmäßig sind und Fehler sonst erst sichtbar werden, wenn sie bereits breiter wirken. Problematisch wird es, wenn normale Schwankungen nicht sauber von echten Fehlern getrennt werden und dadurch unnötige Rücksetzungen ausgelöst werden [21].

Sicherheits- und Bedrohungserkennung umfasst die Anomalieerkennung, Echtzeiterkennung von Angriffen sowie die Klassifikation von Bedrohungen. Ein konkretes Beispiel ist der Einsatz von SL-Algorithmen, um erkannte Anomalien als bösartig 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 [16, 24].

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 Algorithmus, der für alle Anwendungsfälle gleichermaßen geeignet ist. Die Wahl hängt vor allem vom Anwendungsfall, von der verfügbaren Datenbasis und von der Systemarchitektur der Plattform ab.
Am häufigsten werden SL-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 [12].
UL-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 [12].
Etwas weniger thematisiert aber trotzdem relevant sind RL-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 [20, 30].
Bei den Methoden dominieren komplexe Modelle, besonders NN. Das liegt daran, dass sie mit heterogenen Betriebsdaten umgehen können, zum Beispiel mit Protokollen, Metriken und Ereignissen [12]. 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 [34].
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 [4], Jenkins X [16] oder DeepCode [17] genannt, um Build- und Analyseaufgaben zu unterstützen. Für Rollout-Absicherung und Betrieb werden auch Plattformen wie Spinnaker oder Dynatrace [17] beschrieben, die Daten aus dem Betrieb auswerten und Hinweise ableiten. Im

Sicherheitskontext werden Werkzeuge wie Trivy [16], Snyk [33] oder Falco [33] 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

Eine qualitative Bewertung und Priorisierung von KI-Use-Cases ist möglich, wenn KI-Ansätze nicht isoliert betrachtet werden, sondern systematisch in den betrieblichen Kontext eingeordnet werden. Als Ausgangspunkt dient die Einordnung nach Implementierungsaufwand und operativem Mehrwert [18]. Damit lässt sich jedoch nur grob bewerten, ob ein Use Case prinzipiell attraktiv wirkt. Für eine belastbare Aussage zur praktischen Umsetzbarkeit und Wirkung sind ergänzende Dimensionen notwendig.
Die Umsetzbarkeit lässt sich bewerten anhand der verfügbaren Datenqualität und -historie, des technischen Fits zur bestehenden Plattformarchitektur sowie des erforderlichen Fachwissens für Betrieb und Wartung [12, 22]. Die betriebliche Wirksamkeit ergibt sich daraus, ob der Use Case im laufenden Betrieb stabil einen messbaren Nutzen entfaltet und mit vertretbarem Aufwand auf weitere Services übertragbar ist [16, 18]. Ergänzend beeinflussen Anforderungen aus Datenschutz und Governance sowie der Reifegrad der jeweiligen Lösung, ob ein Einsatz realistisch und dauerhaft tragfähig ist [17, 24].
Die qualitative Bewertung entlang einheitlicher Kriterien ermöglicht es, KI-Use-Cases systematisch einzuordnen und miteinander zu vergleichen. Dabei ist nicht entscheidend, welche konkrete KI-Methode oder welches Werkzeug eingesetzt wird. Maßgeblich ist vielmehr, in welchem Verhältnis Implementierungsaufwand, erwarteter Nutzen und betriebliche Rahmenbedingungen zueinander stehen. Die Übertragbarkeit von KI-Lösungen ergibt sich damit nicht aus der Technologie selbst, sondern aus ihrer Passung zur Datenbasis, zu bestehenden Betriebsprozessen und zur organisatorischen Struktur der Plattform.
Die Anwendung des Bewertungskonzepts auf die BMLP zeigt, dass sich generische Use-Case-Muster aus der Literatur auf einen konkreten Plattformkontext übertragen lassen. Use Cases mit vorhandener Datenbasis und klarer Anbindung an bestehende Prozesse lassen sich nachvollziehbar bewerten und priorisieren. Gleichzeitig werden Anwendungsfälle sichtbar, deren Umsetzung zwar theoretisch möglich ist, deren Aufwand oder betriebliche Risiken jedoch aktuell überwiegen. Damit wird eine fundierte Entscheidungsgrundlage geschaffen, die über eine rein werkzeug- oder technologiegetriebene Betrachtung hinausgeht.

Insgesamt zeigt die Arbeit, dass sich identifizierte KI-Lösungen mithilfe des entwickelten Bewertungsschemas strukturiert auf typische Anwendungsfälle im Cloud-Native Platform Engineering übertragen lassen. Die Forschungsfrage RQ3 wird damit beantwortet, indem ein nachvollziehbarer Zusammenhang zwischen KI-Ansätzen, konkreten Plattform-Use-Cases und deren praktischer Umsetzbarkeit hergestellt wird.

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 ein Interpretationsspielraum, 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

Dieses Kapitel fasst die zentralen Ergebnisse der Arbeit zusammen und ordnet sie in den Gesamtkontext ein. Dafür werden zunächst in Kapitel 7.1 die wichtigsten Erkenntnisse zusammengefasst. Der Ausblick in Kapitel 7.2 beschreibt mögliche Ansatzpunkte für weiterführende Forschung und zeigt auf, an welchen Stellen eine vertiefende Untersuchung sinnvoll erscheint.

7.1. Zusammenfassung

Die Arbeit hat gezeigt, dass sich KI-Anwendungen entlang klar abgrenzbarer Anwendungsfelder strukturieren lassen und sich dabei wiederkehrende technische und organisatorische Herausforderungen ergeben. Die Analyse zeigt, dass in der Literatur bislang keine einheitliche Bewertungsgrundlage für KI-Anwendungen im Plattformbetrieb existiert.
Auf Basis einer systematischen Literaturrecherche wurden wiederkehrende Anwendungsfelder und typische Herausforderungen beim Einsatz von KI-Technologien identifiziert. Die Anwendungsfelder betreffen insbesondere Ressourcen- und Workload-Optimierung, den Plattformbetrieb, die Optimierung von CI/CD-Pipelines sowie sicherheitsrelevante Aufgaben. Über alle Felder hinweg zeigen sich ähnliche Herausforderungen, vor allem in den Bereichen Datenqualität und -verfügbarkeit, Integration in bestehende Plattformarchitekturen sowie IT-Sicherheit. Die in der Literatur beschriebenen KI-Ansätze wurden hinsichtlich eingesetzter Werkzeuge, Lernparadigmen und Algorithmen systematisch erfasst und klassifiziert. Darauf aufbauend wurden generische Use-Case-

Muster abgeleitet, die typische Einsatzszenarien und Anforderungen im Plattformbetrieb zusammenfassen.
Für die Bewertung und Auswahl geeigneter KI-Use-Cases wurde ein Bewertungskonzept entwickelt. Dieses ordnet Anwendungsfälle entlang der Dimensionen Implementierungsaufwand und operativer Mehrwert ein. Ergänzend werden Aspekte der Umsetzbarkeit, der betrieblichen Wirksamkeit, der Governance sowie des Reifegrads berücksichtigt. Anhand der Analyse der BMLP wurden wiederkehrende Probleme in Build- und Bereitstellungsprozessen identifiziert. Dieses Problemfeld wurde auf die generischen Use-Case-Muster abgebildet und mithilfe des Bewertungskonzepts analysiert. Auf dieser Grundlage konnten Handlungsempfehlungen einschließlich eines konkreten KI-Tool-Vorschlags abgeleitet werden.

7.2. Ausblick

Der entwickelte Bewertungsansatz ist ein erster Schritt zur Bewertung von KI-Anwendungsfällen. Die Bewertungslogik ist allgemein gehalten und ermöglicht eine qualitative Priorisierung entlang klar definierter Leitfragen. Eine weiterführende Ausarbeitung kann diese Logik durch quantitative Kriterien ergänzen und auf weitere Anwendungsfälle anwenden. Für eine belastbare Entscheidungsgrundlage ist jedoch eine weitere Vertiefung erforderlich. Die Gewichtung der einzelnen Kriterien sollte in zukünftigen Arbeiten systematisch überprüft und kontextabhängig angepasst werden.
Das Bewertungskonzept bietet die Möglichkeit, weitere Anwendungsfälle zu bewerten. Zusätzliche Use Cases lassen sich mit dem gleichen Raster bewerten, ohne die Bewertungslogik grundlegend zu verändern. Dadurch kann das Konzept schrittweise erweitert werden.
Ein nächster Schritt besteht in der prototypischen Umsetzung einzelner priorisierter Anwendungsfälle im Plattformbetrieb. Erst durch den praktischen Einsatz lassen sich Aussagen zur tatsächlichen Wirksamkeit, zum Betriebsaufwand und zu langfristigen Effekten treffen. Dies betrifft vor allem Aspekte wie Stabilität, Wartbarkeit und den Umgang mit veränderlichen Datenlagen.
Insgesamt zeigt die Arbeit, dass eine strukturierte Einordnung von KI im Plattformkontext möglich ist, auch ohne detaillierte Kosten- oder Leistungsmodelle. Dieser Ansatz bietet eine Grundlage, um zukünftige Analysen gezielter und vergleichbarer durchzuführen und damit sowohl Forschung als auch Praxis weiter zu unterstützen.

Tabellenverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

AIOps Artificial Intelligence for IT Operations. 6, 7, 10, 37

BMLP Bosch Digital Manufacturing Platform. i, 2, 3, 12, 29, 36, 37, 39, 47, 51

CD Continuous Delivery. 5, 7, 14, 20, 23, 31, 34, 36, 39, 43, 44, 46, 50
CI Continuous Integration. 5, 7, 14, 20, 23, 26, 31, 34, 36, 39, 43, 44, 46, 50
CNCF Cloud Native Computing Foundation. 4, 7
CPU Central Processing Unit. 19, 22, 24

DL Deep Learning. 6, 17, 18, 21, 24, 27, 28

KI Künstliche Intelligenz. i, 1, 2, 4, 6–15, 17, 19, 20, 23–32, 34–36, 42, 44–48, 50, 51

MLOps Machine Learning Operations. 6, 7, 10

NN Neuronale Netze. 6, 17, 18, 21, 46

RL Reinforcement Learning. 5–7, 16, 21, 24, 25, 46

SL Supervised Learning. 5, 16, 17, 21, 26, 46

UL Unsupervised Learning. 5, 16, 17, 21, 25, 46

A

Anhang

A.1. Suchstrings der Literatursuche

Suchstring 1

```
("Artificial Intelligence" OR "Machine Learning" OR "Deep Learning" OR "AIOps")
AND ("Platform Engineering" OR "Cloud Native" OR "Internal Developer Platform"
OR "DevOps" OR "Platform Operations")
AND ("Automation" OR "Optimization" OR "Developer Experience"
OR "Platform Management")
```

Suchstring 2

```
("Platform Engineering" OR "Cloud Native")
AND ("AI" OR "Machine Learning" OR "AIOps")
```

Suchstring 3

```
("Platform Engineering" OR "DevOps")
AND ("Artificial Intelligence" OR "Machine Learning" OR "AIOps")
```

A.2. Bewertungsmatrix für KI-Anwendungsfälle

Die Python-Implementierung des Bewertungsschemas ist unter https://github.com/ni351arn/Bewertungsmatrix.git dokumentiert.

Literaturverzeichnis

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

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

[3] Steef-Jan Wiggers u. a. *InfoQ Cloud and DevOps Trends Report - 2025*. Okt. 2025. URL: https://www.infoq.com/articles/cloud-devops-trends-2025/ (besucht am 07. 01. 2026).

[4] Adel Zaalouk u. a. *Cloud Native Artificial Intelligence*. CNCF Technical Report. März 2024. URL: https://www.cncf.io/wp-content/uploads/2024/03/cloud_native_ai24_031424a-2.pdf.

[5] Kubernetes Documentation. *Kubernetes Documentation: Concepts - Overview*. URL: https://kubernetes.io/docs/concepts/overview/ (besucht am 15. 11. 2025).

[6] Gerardo Lopez und Saloni Narang. *GitOps in 2025: From Old-School Updates to the Modern Way*. Juni 2025. URL: https://www.cncf.io/blog/2025/06/09/gitops-in-2025-from-old-school-updates-to-the-modern-way/ (besucht am 30. 12. 2025).

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

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

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

[10] Red Hat. *Was ist CI/CD? | Automatisierung in der Softwareentwicklung*. Jan. 2025. URL: https://www.redhat.com/de/topics/devops/what-is-ci-cd (besucht am 16. 12. 2025).

[11] Dhia Elhaq Rzig u. a. *Empirical Analysis on CI/CD Pipeline Evolution in Machine Learning Projects*. März 2024. URL: https://arxiv.org/abs/2403.12199v4.

[12] 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.

[13] 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.

[14] 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.

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

[16] Sudheer u. a. "AI/ML – DevOps Automation". In: *American Journal of Engineering Research (AJER)* 13.10 (Okt. 2024), S. 111–117. ISSN: 2320-0847. URL: https://www.ajer.org/papers/Vol-13-issue-10/1310111117.pdf.

[17] Suprit Pattanayak u. a. "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.

[18] 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.

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

[20] Josson Paul Kalapparambath u. a. "Advancing Distributed Systems with Reinforcement Learning: A New Frontier in AI-Integrated Software Engineering". In: *Sarcouncil Journal of Engineering and Computer Sciences* 10 (März 2025). ISSN: 2945-3585. DOI: 10.5281/zenodo.15305618.

[21] 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: *Distributed Learning and Broad Applications in Scientific Research* 10 (Feb. 2024). ISSN: 2458-1232. URL: https://www.researchgate.net/publication/384804118_AI-Enhanced_Continuous_Integration_and_Continuous_Deployment_Pipelines_Leveraging_Machine_Learning_Models_for_Predictive_Failure_Detection_Automated_Rollbacks_and_Adaptive_Deployment_Strategies_in_Agi.

[22] 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.

[23] 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].

[24] 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: 3050-9726. DOI: 10.54660/.IJFMR.2021.2.2.61-67.

[25] 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 2014, S. 1–10. ISBN: 978-1-4503-2476-2. DOI: 10.1145/2601248.2601268.

[26] Kai Petersen u. a. “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.

[27] Barbara Kitchenham und Stuart M. Charters. *Guidelines for Performing Systematic Literature Reviews in Software Engineering*. Juli 2007. URL: https://www.researchgate.net/publication/302924724%5C_Guidelines%5C_for%5C_performing%5C_Systematic%5C_Literature%5C_Reviews%5C_in%5C_Software%5C_Engineering (besucht am 06. 11. 2025).

[28] Karthik Puthraya u. a. “The Role of Cloud-Native Architectures in Accelerating Machine Learning Workflows through Data Engineering Innovations”. In: *Sarcouncil Journal of Applied Sciences* 5.2 (Feb. 2025). ISSN: 2945-3437. DOI: 10.5281/ZENODO.15106432.

[29] 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.

[30] Gopinath Kathiresan. “Cybersecurity Risk Modeling in CI/CD Pipelines Using Reinforcement Learning for Test Optimization”. In: *International Journal of Innovative Science and Research Technology* 10.5 (Mai 2025), S. 15–25. DOI: 10.38124/ijisrt/25may339.

[31] Rahul Amte. “Cloud-Native AI: Challenges and Innovations in Deploying Large-Scale Machine Learning Models”. In: *International Journal of Scientific Research in Artificial Intelligence and Machine Learning* 6.2 (März 2025), S. 9–18. ISSN: 3067-753X. DOI: 10.63397/ISCSITR-IJSRAIML_06_02_002.

[32] 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. URL: https://journalofemergingtechnologyanddigitaltransformation.com/index.php/3/article/view/109.

[33] 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). ISSN: 2321-8169. DOI: DOI:10.5281/zenodo.14662301. URL: https://www.researchgate.net/publication/388036382_Cloud-Native_Application_Development_for_AI-_Conducive_Architectures.

[34] Vijay Govindarajan. “Machine Learning Based Approach for Handling Imbalanced Data for Intrusion Detection in the Cloud Environment”. In: *Proceedings of the 3rd International Conference on Disruptive Technologies (ICDT)*. März 2025, S. 810–815. DOI: 10.1109/ICDT63985.2025.10986614. URL: https://ieeexplore.ieee.org/document/10986614.