Fachhochschule der Wirtschaft
-FHDW-
Paderborn

Bachelorthesis

Thema:

Effiziente Workload-Automatisierung durch Self-Service und Künstliche Intelligenz: Möglichkeiten bei Streamworks

Prüfer:
<< Erstprüfer >>
<< Zweitprüfer >>

Verfasser:
Ravel-Lukas Geck
Lagesche Str. 258
32758 Detmold

Wirtschaftsinformatik
Business Process Management

Eingereicht am:

20.10.2025

Inhaltsverzeichnis

1 Einleitung

1.1 Problemstellung und Motivation

Die Automatisierung von Enterprise-Workloads hat sich zu einem kritischen Erfolgsfaktor moderner IT-Landschaften entwickelt. Während viele Unternehmen bereits Fortschritte bei der Standardisierung ihrer Geschäftsprozesse erzielt haben, bleibt der Zugang zu leistungsstarken Workload-Management-Systemen wie Streamworks auf einen kleinen Kreis von Experten beschränkt. Diese Zugangsbarriere führt zu Engpässen und verhindert die effiziente Nutzung vorhandener Automatisierungspotenziale.

Streamworks-Expertensystem ohne Self-Service-Optionen

Streamworks basiert auf einem GUI-basierten Baukasten-Prinzip mit Desktop-Client-Oberfläche, die sich in fünf Bereiche gliedert: Dashboard, Administration, Stream-Design, Runtime und Reporting. Experten können über grafische Oberflächen komplexe Workload-Konfigurationen erstellen und Streams visuell als Jobnetze darstellen. Für Fachanwender aus anderen Abteilungen existieren jedoch keine Self-Service-Optionen.

Jede neue Workload-Anforderung durchläuft denselben aufwändigen Prozess. Fachabteilungen formulieren Anforderungen in natürlicher Sprache wie "Wir benötigen einen täglichen Export der SAP-Daten nach CSV", die von Streamworks-Experten in GUI-Konfigurationen übersetzt werden müssen. Hochqualifizierte Experten werden so für repetitive Konfigurationsaufgaben eingesetzt, obwohl sich viele Anfragen auf ähnliche Muster reduzieren lassen.

Experteninterviews bestätigen diese Problematik. Ein erfahrener Streamworks-Spezialist betont: "Eine KI versteht nie, was du sagst... sobald es Richtung Kunden geht oder sobald es Richtung Umsetzung geht, muss immer ein Mensch darüber schauen." Bei Standard-Jobs muss häufig "nicht noch einer rüber gucken" - ein Hinweis darauf, dass ein erheblicher Teil der Workload-Erstellung standardisierbaren Mustern folgt.

XML-Export als Automatisierungsschnittstelle

Eine interessante Möglichkeit zur Überbrückung dieser Zugangsbarriere bietet das XML-Export/Import-System von Streamworks. Während die normale Konfiguration über die GUI erfolgt, können "sämtliche Prozess-Definitionen über das Streamworks Export/Import Utility im XML-Format transferiert werden." Dieses XML-Format könnte als Automatisierungsschnittstelle dienen.

Moderne Large Language Models zeigen beeindruckende Fähigkeiten bei der Übersetzung zwischen natürlichsprachlichen Beschreibungen und strukturierten Formaten. Ein KI-gestütztes Self-Service-System könnte natürlichsprachliche Anforderungen in entsprechende XML-Definitionen übersetzen, die dann automatisch importiert werden. Der Experte behält dabei die finale Kontrolle über Import-Prozess und Validierung.

Die vorliegende Arbeit untersucht die technische Machbarkeit einer solchen Lösung. Das Ziel ist ein funktionsfähiger Prototyp, der demonstriert, wie KI-Technologien die Zugangsbarrieren zu Streamworks überwinden können, ohne die bewährte Expertenkonfiguration zu ersetzen, sondern diese um Self-Service-Möglichkeiten zu erweitern.

1.2 Zielsetzung der Arbeit

Das primäre Ziel dieser Bachelorarbeit ist die Entwicklung eines hybriden Self-Service-Systems für Streamworks-Konfigurationen durch Large Language Models. Eine benutzerfreundliche Web-Oberfläche soll Fachanwendern ermöglichen, komplexe Workload-Streams durch natürlichsprachige Eingaben zu definieren, ohne tiefgreifende Streamworks-Kenntnisse zu benötigen.

Hauptziel: Automatisierte XML-Stream-Erstellung

Das System soll natürlichsprachliche Anforderungen wie "Erstelle einen täglichen SAP-Export der Kundendaten nach CSV" automatisch in entsprechende XML-Definitionen übersetzen, die über das Streamworks Import-Utility eingelesen werden können. Dabei wird die Komplexität der XML-Generierung vor dem Endbenutzer verborgen, während die Flexibilität und Mächtigkeit der Streamworks-Plattform vollständig erhalten bleibt. Durch die Integration von LLM-Technologien entsteht eine natürliche Schnittstelle zwischen menschlicher Intention und maschinenlesbarer Konfiguration.

Nebenziel: RAG-basiertes Support-System

Als ergänzende Komponente wird ein Retrieval-Augmented Generation-basiertes Q&A-System entwickelt, das als intelligente Dokumentations- und Support-Plattform für Streamworks-Nutzer fungiert. Dieses System macht vorhandene Dokumentationen, Best Practices und Konfigurationsbeispiele durchsuchbar und bietet kontextabhängige Hilfestellungen bei der Stream-Erstellung.

Messbare Erfolgsparameter

Die Arbeit verfolgt konkrete, quantifizierbare Ziele: 60% Zeitersparnis bei der Stream-Erstellung gegenüber manueller Konfiguration, Reduzierung der benötigten Streamworks-Expertise um mindestens 80% für Standard-Workload-Erstellungen, sowie eine

Fehlerrate unter 5% bei der automatischen XML-Generierung. Das System soll mindestens 70% der häufigsten Streamworks-Use-Cases wie SAP-Jobs, Dateitransfers und Monitoring-Streams vollautomatisch generieren können, während das RAG-System mindestens 80% der Standardfragen ohne menschliche Intervention beantworten soll

Wissenschaftlicher Beitrag

Diese Arbeit leistet einen Beitrag zur Forschung im Bereich Enterprise-Automatisierung durch KI durch die praktische Anwendung von LLMs für domänenspezifische XML-Generierung in produktiven Enterprise-Umgebungen und die Entwicklung einer hybriden Architektur zur Balancierung von Automatisierung und menschlicher Kontrolle bei kritischen Geschäftsprozessen.

Projektumfang und Abgrenzung

Der Scope umfasst die Entwicklung eines Self-Service-Portals für XML-Stream-Generierung, LLM-Integration für natürlichsprachige Konfigurationserstellung, XSD-Validierung und Qualitätssicherung sowie ein RAG-basiertes Q&A-System für Streamworks-Support. Außerhalb des Scopes liegen produktive Deployment-Strategien für Enterprise-Umgebungen, vollständige Integration in bestehende Identity-Management-Systeme sowie umfassende Streamworks-API-Integration über XML-Import/Export hinaus.

Das Ergebnis ist ein funktionsfähiger Prototyp, der die technische Machbarkeit der LLM-basierten XML-Generierung für Enterprise-Workload-Management demonstriert und eine strategische Roadmap für die Weiterentwicklung und produktive Einführung bietet.

1.3 Methodisches Vorgehen

Diese Bachelorarbeit folgt einem hybriden Forschungsansatz, der theoretische Fundierung mit praktischer Implementierung kombiniert. Die Methodik gliedert sich in vier aufeinander aufbauende Phasen.

Literaturrecherche und theoretische Fundierung

Systematische Literaturrecherche zu Workload-Automatisierung, Self-Service-Konzepten, Large Language Models und Retrieval-Augmented Generation. Auswertung wissenschaftlicher Publikationen, Industriestandards und technischer Dokumentationen mit Fokus auf aktuelle Entwicklungen 2020-2025 über IEEE Xplore und ACM Digital Library.

Experteninterviews und Anforderungsanalyse

Strukturierte Experteninterviews mit Streamworks-Spezialisten zur Validierung theoretischer Erkenntnisse und Identifikation tatsächlicher Problemstellungen. Semi-

strukturierter Ansatz mit standardisierten Fragen und explorativen Nachfragen zu typischen Anwendungsfällen und organisatorischen Rahmenbedingungen.

Prototypische Entwicklung

Agile Implementierung in iterativen Zyklen, beginnend mit Minimum Viable Product. Entwicklung von Frontend-Komponenten für Self-Service-Portal, Backend-Services für LLM-Integration sowie Validierungs- und Import-Mechanismen für XML-Generierung.

Evaluation und Bewertung

Quantitative Metriken: Generierungsgeschwindigkeit, Fehlerrate der XML-Ausgabe, Validierungsergebnisse gegen Streamworks-Schema. Qualitative Bewertung durch Usability-Tests mit potenziellen Anwendern und Feedback von Streamworks-Experten.

Die Kombination aus theoretischer Fundierung, praktischer Implementierung und empirischer Evaluation ermöglicht eine umfassende Bearbeitung der Forschungsfrage bei gleichzeitiger Erfüllung wissenschaftlicher und praktischer Anforderungen.

1.4 Aufbau der Arbeit

Diese Bachelorarbeit folgt einer logischen Progression von der Problemanalyse über die theoretische Fundierung zur praktischen Implementierung und kritischen Bewertung

Kapitel 2 legt die theoretischen Grundlagen durch vier zentrale Themenbereiche: Workload-Automatisierung in Enterprise-Umgebungen, Self-Service-Konzepte in der IT, Large Language Models und Retrieval-Augmented Generation. Diese Kapitel definieren den wissenschaftlichen Rahmen und aktuelle Herausforderungen.

Kapitel 3 analysiert die spezifische Streamworks-Umgebung mit Architektur, aktuellen Prozessen, Automatisierungspotenzialen und Anforderungen an Self-Service-Lösungen.

Kapitel 4 entwickelt das Konzept und Design der Lösung, umfassend Systemarchitektur, Self-Service-Portal, automatisierte XML-Generierung und RAG-Integration.

Kapitel 5 dokumentiert die konkrete Implementierung des Self-Service-Frontend, der LLM-Integration, XSD-Validierung und des RAG-basierten Q&A-Systems.

Kapitel 6 evaluiert das entwickelte System durch Testmethodologie, Qualitätsbewertung, Effizienzsteigerung und Bewertung des Support-Systems.

Kapitel 7 präsentiert die Ergebnisse bezüglich Funktionalität, Performance, Benutzerfreundlichkeit und Mehrwert des Q&A-Systems.

Kapitel 8 reflektiert kritisch die Limitationen der LLM-basierten XML-Generierung, Herausforderungen bei der Validierung und Skalierbarkeit.

Kapitel 9 schließt mit einer Zusammenfassung der Ergebnisse, dem Beitrag zur Streamworks-Automatisierung und zukünftigen Entwicklungsmöglichkeiten ab.

Diese Gliederung gewährleistet eine systematische Herangehensweise und verbindet wissenschaftliche Rigorosität mit praktischer Relevanz.

2 Theoretische Grundlagen

Die zunehmende Digitalisierung von Unternehmen erfordert eine strategische Ausrichtung des Informationsmanagements. Insbesondere die prozessorientierte Gestaltung von IT-Infrastrukturen spielt eine zentrale Rolle, um Geschäftsprozesse effizient zu unterstützen, zu steuern und zu automatisieren[1]. Dieses Kapitel legt die theoretischen Grundlagen für die Entwicklung eines KI-gestützten Self-Service-Systems zur Workload Automation.

Zunächst wird in Kapitel 2.1 die Workload Automation als etabliertes Konzept der IT-Prozessautomatisierung eingeführt. Aufbauend darauf behandelt Kapitel 2.2 das Self-Service-Paradigma, welches Nutzern ermöglicht, eigenständig auf IT-Ressourcen und -Services zuzugreifen.

Die Integration künstlicher Intelligenz in Form von Large Language Models (LLMs), dargestellt in Kapitel 2.3, eröffnet neue Möglichkeiten für intuitive, natürlichsprachliche Interaktionen mit technischen Systemen. Abschließend thematisiert Kapitel 2.4 das Konzept der Retrieval Augmented Generation (RAG), dass die Leistungsfähigkeit großer Sprachmodelle durch die gezielte Einbindung externer oder interner Datenquellen erheblich erweitert. RAG ermöglicht eine kontextspezifischere und verlässlichere Antwortgenerierung, indem relevante Informationen aus verknüpften Datenbanken, Dokumentensammlungen oder Wissensgraphen dynamisch in den Prompt integriert werden – ganz ohne Fine-Tuning.[2]

Diese vier Komponenten bilden zusammen das theoretische Fundament für die in dieser Arbeit konzipierte Lösung eines intelligenten Self-Service-Systems für Workload Automation.

2.1 Workload-Automatisierung in Enterprise-Umgebungen

Workload-Automatisierung (WLA) bezeichnet die IT-gestützte Planung, Steuerung und Kontrolle regelbasierter Abläufe und Routineprozesse[3]. Als integraler Bestandteil moderner IT-Infrastrukturen reduziert sie manuelle Eingriffe und erhöht die Prozessqualität.

---

[1] Vgl. Heinrich et al. (2020), *Informationsmanagement – Grundlagen, Aufgaben, Methoden*, S. 6

[2] Vgl. Honroth et al. (2024), *Retrieval Augmented Generation (RAG): Chat mit eigenen Daten*, Fraunhofer IESE, S.3

[3] Vgl. Gadatsch (2020), *Grundkurs Geschäftsprozessmanagement*, S. 16–18

2.1.1 Definition und Abgrenzung

Workload-Automatisierung (WLA) bezeichnet den Einsatz spezialisierter Software, um Geschäfts- und IT-Prozesse durch zeit- oder ereignisgesteuerte Abläufe automatisiert zu steuern und zu überwachen. Sie ermöglicht die zentrale Orchestrierung von Workloads über heterogene Systeme und Plattformen hinweg und reduziert dadurch manuelle Eingriffe sowie Fehleranfälligkeit[4]. Im Gegensatz zu reiner Jobausführung integriert WLA Abhängigkeiten, Ereignisse und Geschäftskontexte, wodurch sie eine entscheidende Grundlage für die digitale Unternehmenssteuerung bildet[5].

Die Abgrenzung zu Job Scheduling liegt vor allem im Reifegrad: Klassische Scheduler arbeiten zeit- oder batchorientiert und sind meist auf einzelne Systeme beschränkt. Workload-Automatisierung erweitert dieses Konzept um zentrale Steuerung, ereignisgesteuerte Trigger und plattformübergreifende Integration, wodurch komplexe, unternehmensweite Prozesse beherrschbar werden[6].

Von Robotic Process Automation (RPA) unterscheidet sich WLA dadurch, dass RPA vorrangig regelbasierte, repetitive Tätigkeiten auf der Benutzerschnittstelle nachahmt, wie etwa Dateneingaben in ERP- oder CRM-Systeme. WLA hingegen steuert ganze Prozessketten auf Infrastrukturebene und integriert verschiedene Anwendungen im Backend[7].

Business Process Management (BPM) verfolgt einen umfassenderen Ansatz: Ziel ist die Modellierung, Analyse und Optimierung von End-to-End-Prozessen auf organisatorischer Ebene. Während BPM den methodischen Rahmen und die Prozesslogik definiert, übernimmt WLA die technische Umsetzung durch operative Automatisierungsschritte[8].

Die Enterprise-Relevanz von Workload-Automatisierung zeigt sich darin, dass Unternehmen zunehmend hybride IT-Landschaften orchestrieren müssen. Studien belegen, dass moderne WLA-Lösungen zu mehr Effizienz, Zuverlässigkeit und Transparenz führen und in der Mehrheit der Organisationen als strategisches Fundament für digitale Transformation gelten[9].

---

[4] Vgl. Sabharwal/Kasiviswanathan, *Introduction to Workload Automation*, S. 1.

[5] Vgl. Redwood, *Workload Automation – What It Is And Why Enterprises Use It*, S. 1.

[6] Vgl. Chandio et al., *A Comparative Study of Job Scheduling Strategies in Large-Scale Parallel Computational Systems*, S. 2-3

[7] Vgl. Aguirre/Rodriguez, *Automation of a Business Process Using Robotic Process Automation (RPA)*, S. 6

[8] Vgl. Mendling et al., *Fundamentals of Business Process Management*, S. 1-3

[9] Vgl. Stonebranch, *2025 Global State of IT Automation Report*, S. 6.

2.1.2 Historische Entwicklung

Die Entwicklung der Workload-Automatisierung (WLA) lässt sich über mehrere Jahrzehnte hinweg nachzeichnen und verdeutlicht den engen Zusammenhang zwischen technologischen Umbrüchen und Automatisierungslösungen.

In den 1970er Jahren dominierten zunächst einfache Task- und Batch-Scheduler, die in zentralisierten Mainframe-Umgebungen zeitgesteuerte Aufgaben abwickelten. Diese frühen Systeme dienten vor allem der sequentiellen Abarbeitung von Jobs und bildeten den Grundstein für spätere Automatisierungslösungen[10].

Mit der Dezentralisierung der IT-Landschaften in den 1980er Jahren erhöhte sich die Komplexität. Scheduling-Werkzeuge wurden um Funktionen wie die Koordination über verteilte Systeme und die Durchführung von Managed File Transfers erweitert. In den 1990er Jahren kam mit dem Aufstieg des Internets eine weitere Dimension hinzu, die neue Einsatzmöglichkeiten eröffnete und den Markt für Job Scheduling deutlich ausweitete[11].

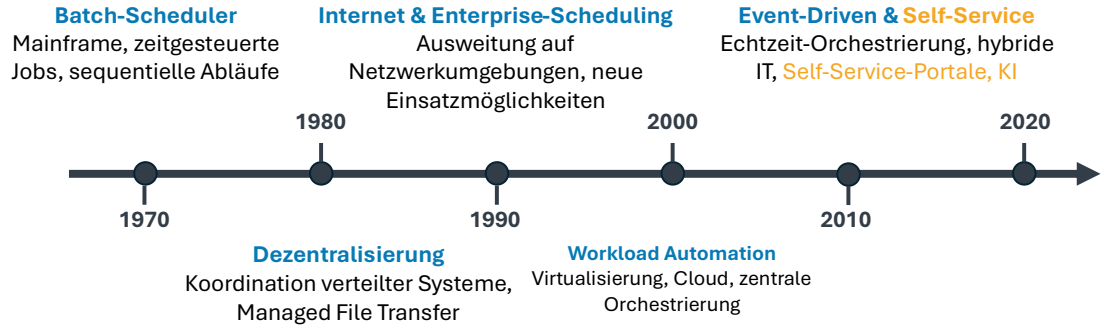


Abbildung 1: Entwicklung der Workload Automatisierung (Eigene Darstellung)

Ab den 2000er Jahren setzte sich der Begriff Workload Automation durch. Treiber hierfür waren Virtualisierung, Cloud-Computing und E-Business-Anwendungen, die eine zentrale Steuerung heterogener Systeme notwendig machten. WLA-Lösungen wurden zunehmend mit IT-Service-Management- und IT-Prozess-Automatisierungstools integriert, wodurch sie einen übergreifenden Ansatz zur Orchestrierung von IT-Prozessen boten[12].

In den 2010er Jahren galt WLA als etablierte Praxis, jedoch verlangsamte sich die Innovationsgeschwindigkeit. Erst mit dem Siegeszug von Big Data und Cloud-Technologien gegen Ende des Jahrzehnts erhielt das Feld neue Impulse. WLA-Systeme wurden

[10] Vgl. *Workload Automation: History Through the Decades*, S. 1
[11] Vgl. *Workload Automation: History Through the Decades*, S. 1–2
[12] Vgl. *Workload Automation: History Through the Decades*, S. 1-2

ausgeweitet, um große Datenmengen, verteilte Anwendungen und hybride IT-Umgebungen abzudecken[13].

Die jüngsten Entwicklungen der 2020er Jahre zeigen eine weitere Evolution hin zu ereignisgesteuerten Service-Orchestration- und Automatisierungsplattformen (SOAP). Diese modernen Systeme ermöglichen nicht nur klassische Job-Scheduling-Aufgaben, sondern orchestrieren komplexe Workflows in Echtzeit, integrieren Cloud- und Container-Infrastrukturen und bieten Self-Service-Funktionalitäten für Fachbereiche. Damit verschiebt sich der Fokus von rein technischer Steuerung hin zur ganzheitlichen Unternehmensautomatisierung[14].

2.1.3 Kernfunktionen moderner WLA

Moderne Workload-Automatisierung (WLA) zeichnet sich durch eine Reihe zentraler Kernfunktionen aus, die den Betrieb komplexer IT-Landschaften erheblich effizienter und sicherer gestalten. Im Mittelpunkt steht die zentrale Steuerung von Jobs und Prozessen über eine einheitliche Plattform, die es ermöglicht, heterogene Systeme und Anwendungen konsolidiert zu orchestrieren und dadurch die Transparenz sowie die Betriebssicherheit zu erhöhen[15].

Ein wesentliches Merkmal ist das Event-Driven Scheduling, bei dem Prozesse nicht ausschließlich zeitgesteuert, sondern dynamisch auf Ereignisse reagieren. Dies erlaubt eine nahezu in Echtzeit erfolgende Ausführung von Workflows, was insbesondere in hochvolumigen Umgebungen eine erhöhte Agilität und Ressourcenauslastung ermöglicht[16]. Eng damit verbunden ist das Abhängigkeitsmanagement, das die automatische Berücksichtigung von Vorbedingungen und Verknüpfungen zwischen Prozessen sicherstellt. So wird gewährleistet, dass nachgelagerte Aufgaben erst dann ausgeführt werden, wenn alle relevanten Vorgänger erfolgreich abgeschlossen wurden, wodurch Fehler und Inkonsistenzen minimiert werden[17].

---

[13] Vgl. *Workload Automation: History Through the Decades*, S. 2

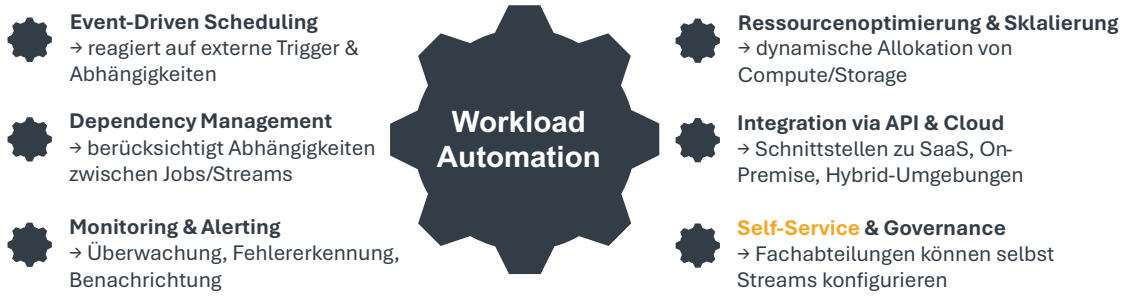
[14] Vgl. *Stonebranch – 2025 Global State of IT Automation Report*, S. 6–7; 16

[15] Vgl. IT Central Station, *Selecting a Workload Automation Tool*, S. 2

[16] Vgl. Talaseela, *Intelligent Workflow Orchestration for Enterprise Contexts*, S. 27–29

[17] Vgl. Dhanaraj/Banu, *A Taxonomy and Survey of Scheduling Algorithms in Cloud: based on Task Dependency*, S. -3

Zur Aufrechterhaltung von Stabilität und Nachvollziehbarkeit kommt dem Monitoring eine zentrale Rolle zu. Moderne Systeme bieten umfassende Überwachungs- und Benach-



richtigungsfunktionen, die es erlauben, Störungen frühzeitig zu erkennen und proaktiv einzugreifen. Ergänzend hierzu ist die API-Integration von strategischer Bedeutung, da sie eine nahtlose Anbindung externer Systeme ermöglicht und den Austausch von Daten sowie die Steuerung von Prozessen über standardisierte Schnittstellen sicherstellt[18].

Darüber hinaus hat sich in den letzten Jahren der Trend zu Self-Service-Funktionalitäten etabliert. Diese befähigen Fachabteilungen, eigenständig Automatisierungsprozesse zu initiieren, ohne auf die IT-Abteilung angewiesen zu sein. In Verbindung mit Role-Based Access Control (RBAC) wird dabei sichergestellt, dass Zugriffsrechte strikt nach Rollen und Verantwortlichkeiten vergeben werden, wodurch Sicherheit und Governance-Anforderungen gewahrt bleiben[19].

Zusammenfassend lässt sich festhalten, dass moderne WLA-Lösungen durch diese Kernfunktionen nicht nur operative Effizienz steigern, sondern auch ein höheres Maß an Flexibilität, Transparenz und Sicherheit in zunehmend komplexen IT-Infrastrukturen gewährleisten.

2.1.4 Aktuelle Herausforderungen und Trends

Die Workload-Automatisierung (WLA) steht aktuell vor einer Reihe von Herausforderungen, die sowohl technologische als auch organisatorische Dimensionen umfassen. Ein zentrales Problem bildet die zunehmende Komplexität hybrider IT-Landschaften. Mit rund 77 % der Unternehmen, die hybride Infrastrukturen betreiben, gewinnt die Orchestrierung zwischen On-Premises-, Cloud- und containerisierten Systemen an kritischer Bedeutung[20]. Die Integration heterogener Systeme erfordert dabei leistungsfähige

[18] vgl. IBM, *Driving IBM Z Workload Scheduler with REST API*, S. 1

[19] Vgl. Sandhu et al., *The NIST Model for Role-Based Access Control: Towards a Unified Standard*, S. 1-2

[20] Vgl. Stonebranch, *2025 Global State of IT Automation Report*, S. 7

Schnittstellen, um Fragmentierung und „Tool Sprawl“ zu vermeiden[21]. Hinzu kommen Sicherheits- und Compliance-Anforderungen, die insbesondere in Public-Cloud-Umgebungen als wesentliches Hemmnis wahrgenommen werden[22]. Neben diesen technologischen Hürden verschärft der Fachkräftemangel die Lage, da fehlende Expertise in Automatisierung und Cloud-Technologien die Skalierung entsprechender Initiativen erschwert[23].

Parallel zu diesen Herausforderungen zeichnen sich jedoch deutliche Trends ab. Event-driven Automation gewinnt an Bedeutung, da sie eine flexiblere, ereignisgesteuerte Steuerung von Prozessen ermöglicht und so die starre Batch-Verarbeitung ergänzt[24]. Zudem verschiebt sich der Fokus auf KI-gestützte Orchestrierung, die dynamische Ressourcenzuweisungen und selbstheilende Workflows erlaubt[25]. Besonders hervorzuheben ist der Trend zur Self-Service-Automation, die von einer wachsend breiten Nutzerbasis – auch außerhalb klassischer IT-Abteilungen – getragen wird. IT-Teams entwickeln sich zunehmend zu Anbietern von „Automation-as-a-Service“, wodurch Fachabteilungen befähigt werden, Automatisierungen eigenständig umzusetzen[26]. Ergänzend gewinnt das Konzept der Human-Centered Automation (HCA) an Relevanz, das eine stärkere Ausrichtung der Systeme an Nutzerbedürfnissen verfolgt. Ziel ist es, durch intuitive Oberflächen und geringere Einstiegshürden die Demokratisierung von Automatisierung zu fördern und so die Akzeptanz bei nicht-technischen Anwendern zu steigern[27].

Damit wird deutlich, dass die Workload-Automatisierung gleichermaßen vor komplexen Integrations- und Governance-Aufgaben steht, während sie sich gleichzeitig in Richtung einer stärker nutzerzentrierten, KI-gestützten und self-service-orientierten Zukunft entwickelt.

---

[21] Vgl. Stonebranch, *2025 Global State of IT Automation Report*, S. 6, 21
[22] Stonebranch, *2025 Global State of IT Automation Report*, S. 12
[23] EMA Research, *The Future of Workload Automation and Orchestration* (2025), S. 2
[24] EMA Research, *The Future of Workload Automation and Orchestration* (2025), S. 7–8
[25] EMA Research, *The Future of Workload Automation and Orchestration* (2025), S. 2
[26] Stonebranch, *2025 Global State of IT Automation Report*, S. 7
[27] Vgl. Toxtli, *Human-Centered Automation* (2024), S. 1–3

2.2 Self-Service-Konzepte in der IT-Administration

2.2.1 Definition von Self-Service, Prosumer und Human-Centered Automation (HCA)

Der Begriff Self-Service bezeichnet im Unternehmenskontext technologische Lösungen, die es Nutzern ermöglichen, Dienstleistungen eigenständig in Anspruch zu nehmen, ohne dass eine direkte Interaktion mit einem Servicemitarbeiter erforderlich ist. Solche Self-Service-Technologien (SST) umfassen beispielsweise Online-Banking, Self-Check-in am Flughafen oder interne IT-Self-Service-Portale. Sie tragen dazu bei, Abläufe effizienter zu gestalten und gleichzeitig die Abhängigkeit von Servicepersonal zu reduzieren. Studien zeigen, dass SSTs für Unternehmen insbesondere zur Kostensenkung, Effizienzsteigerung und Standardisierung von Serviceprozessen beitragen, während Nutzende vor allem von höherer Flexibilität, Zeitersparnis und gesteigerter Autonomie profitieren[28].

Im Zusammenhang mit Self-Service ist der Begriff Prosumer zentral. Er geht auf Toffler zurück und beschreibt die Verschmelzung von Produzent und Konsument. Prosumers sind nicht mehr passive Abnehmer von Dienstleistungen, sondern übernehmen aktiv eine Rolle in der Wertschöpfung. Sie gestalten Serviceprozesse mit, indem sie mittels Technologie Leistungen eigenständig produzieren, anpassen und konsumieren. Damit wird der Konsument zum „Ko-Produzenten“ und trägt unmittelbar zur Effizienz und Individualisierung der Servicebereitstellung bei[29].

Ergänzend dazu stellt Human-Centered Automation (HCA) einen konzeptionellen Ansatz dar, der den Menschen bewusst in den Mittelpunkt von Automatisierungsprozessen rückt. Im Gegensatz zu rein technologiegetriebenen Automatisierungslösungen zielt HCA darauf ab, Transparenz, Steuerbarkeit und Vertrauen in automatisierte Systeme zu gewährleisten. Nutzer sollen nicht ersetzt, sondern unterstützt werden, indem Automatisierung so gestaltet wird, dass menschliche Stärken – wie Urteilsfähigkeit und Kontextbewusstsein – mit den Vorteilen maschineller Effizienz kombiniert werden. Damit wird ein Gleichgewicht zwischen Automatisierung und menschlicher Kontrolle angestrebt[30].

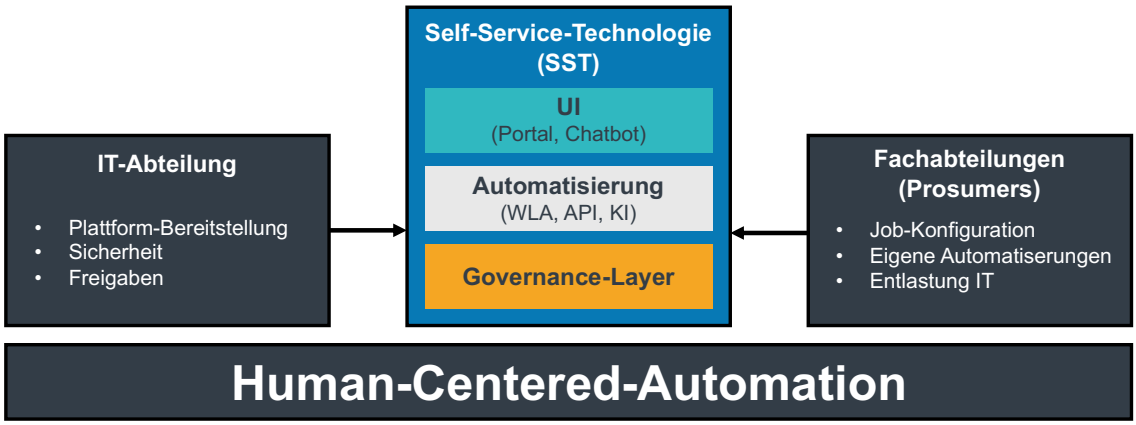
---

[28] Vgl. Considine/Cormican, *The rise of the prosumer: an analysis of self-service technology adoption in a corporate context*, S. 26–28

[29] Vgl. Considine/Cormican, *The rise of the prosumer: an analysis of self-service technology adoption in a corporate context*, S. 26

[30] Vgl. Sheridan, *Human-Centered Automation*, S. 1–2

Zusammenfassend verdeutlichen die drei Begriffe die zentrale Rolle des Nutzers in modernen Automatisierungsszenarien: Während Self-Service die operative Autonomie stärkt und Prosumer die aktive Teilhabe in Wertschöpfungsprozessen betont, stellt HCA sicher, dass Automatisierung nutzerorientiert, transparent und unterstützend ausgestaltet ist.



2.2.2 Charakteristika und technologische Grundlagen von Self-Service-Systemen

Self-Service-Systeme im Kontext des IT-Service-Managements (ITSM) zeichnen sich durch eine Reihe zentraler Charakteristika aus. Ein wesentliches Merkmal ist ihre Benutzerfreundlichkeit, da Mitarbeiter in der Lage sein sollen, eigenständig IT-Services zu bestellen oder Änderungen vorzunehmen, ohne auf manuelle Unterstützung durch IT-Abteilungen angewiesen zu sein[31]. Damit verbunden ist eine hohe Standardisierung, die sowohl in strukturierten Service-Katalogen als auch in vordefinierten Request-Typen Ausdruck findet und eine konsistente Abwicklung ermöglicht[32]. Zudem weisen Self-Service-Portale ein erhebliches Automatisierungspotenzial auf, indem wiederkehrende Routineaufgaben systematisch durch Workflow-Engines) unterstützt werden[33]. Diese Eigenschaft trägt unmittelbar zur Skalierbarkeit bei, da eine steigende Anzahl von Serviceanfragen ohne proportionalen Mehraufwand bearbeitet werden kann[34]. Schließlich spielt auch Governance eine zentrale Rolle: Die Transparenz in Prozessen und die Nachvollziehbarkeit von Service-Requests sind entscheidend, um Compliance-Vorgaben einzuhalten und eine effiziente Ressourcennutzung sicherzustellen[35].

---

[31] Vgl. Floerecke, *Self-Service-Portale mit Schwerpunkt Service-Request-Management*, S. 684
[32] Vgl. Floerecke, *Self-Service-Portale für das IT-Service-Request-Management*, S. 1091f
[33] Vgl. Bock/Frank, *Low-Code Platform*, S. 734-736
[34] Vgl. Floerecke, *Self-Service-Portale mit Schwerpunkt Service-Request-Management, S. 684f*
[35] Vgl. Floerecke, *Self-Service-Portale für das IT-Service-Request-Management*, S. 1090

Die technologische Grundlage von Self-Service-Systemen stützt sich auf mehrere Ebenen. Im Frontend stellen Benutzeroberflächen wie Webportale, mobile Apps oder zunehmend auch Chatbots die Interaktionsebene dar. Im Backend erfolgt die Integration in bestehende ITSM-Suiten sowie die Anbindung an Datenbanken und Schnittstellen, wodurch ein durchgängiger Informationsfluss gewährleistet wir. Automatisierungs-Engines bilden das Herzstück, da sie Service-Workflows abbilden und mit Workload-Automatisierung (WLA) oder RPA-Lösungen verknüpft werden können. Ergänzend dazu sind Monitoring- und Logging-Funktionen essenziell, um die Nutzung der Portale zu überwachen, Prozesskennzahlen zu erheben und Fehlerquellen frühzeitig zu identifizieren[36]. Schließlich sichern Sicherheitsmechanismen wie rollenbasierte Zugriffskontrollen (RBAC) und Identity-Management-Systeme die Integrität und Vertraulichkeit sensibler Daten ab[37].

Insgesamt wird deutlich, dass die technologische Architektur die zuvor genannten Charakteristika maßgeblich ermöglicht: Benutzerfreundliche Oberflächen schaffen Akzeptanz, Automatisierungs-Engines realisieren Effizienzgewinne, während Sicherheits- und Governance-Mechanismen die notwendige Kontrolle gewährleisten.

2.2.3 Erfolgsfaktoren und Herausforderungen von Self-Service-Systemen

Die erfolgreiche Einführung und Nutzung von Self-Service-Systemen hängt maßgeblich von klar definierten Erfolgsfaktoren ab. Ein zentrales Element ist die Etablierung eindeutiger Governance-Strukturen und Richtlinien, die Verantwortlichkeiten sowie Prozesse festlegen und dadurch die notwendige Transparenz schaffen[38]. Ebenso entscheidend ist eine nutzerzentrierte Gestaltung: Studien zeigen, dass die wahrgenommene Nützlichkeit eines Portals stärker auf die Akzeptanz wirkt als dessen reine Bedienfreundlichkeit, wodurch ein klarer Mehrwert für den Anwender erlebbar sein muss[39]. Neben dem Design ist auch die Integration in bestehende Geschäftsprozesse und Systeme essenziell. Hierbei erweist sich insbesondere die enge Zusammenarbeit zwischen IT, Fachbereichen und Endnutzern als kritischer Erfolgsfaktor, um sowohl technische Anforderungen als auch Nutzerbedürfnisse abzudecken[40]. Darüber hinaus sind kontinuierliche Schulungen

---

[36] Vgl. Kyberna (2025) – Wissensartikel Self Service Portale, S. 3f

[37] Vgl. Floerecke, *Self-Service-Portale für das IT-Service-Request-Management*, S. 1092

[38] Vgl. Floerecke, Best-Practices für die Gestaltung von IT-Service-Katalogen und den Einsatz von Self-Service-Portalen, S. 1512

[39] Vgl. Hartmann/Kerssenfischer/Fritsch/Nguyen, User Acceptance of Customer Self-Service Portals, S. 150

[40] Vgl. Mauludina/Mulyani/Adrianto, *Critical Success Factors for Implementation of Self-Service Business Intelligence in Management Accounting*, S. 299

und begleitendes Change Management erforderlich, um Akzeptanzbarrieren zu reduzieren und die Eigenständigkeit der Nutzer im Umgang mit den Systemen zu fördern. Schließlich muss die Berücksichtigung von Sicherheits- und Compliance-Aspekten bereits zu Beginn der Implementierung erfolgen, um regulatorische Anforderungen einzuhalten und Vertrauen in das System zu schaffen[41].

Erfolgsfaktoren	Herausforderungen
• Klare Governance & Richtlinien • Nutzerzentrierung (Wahrgenommener Nutzen) • Integration in bestehende Prozesse • Schulungen & Change Management • Sicherheit & Compliance	• Schatten-IT • Aktzeptanzprobleme (Innovationsangst) • Technologische Komplexität (Schnittstellen, Datenquellen) • Kontinuierliche Anpassung (Technologie-Fortschritt)

Den Erfolgsfaktoren stehen typische Herausforderungen gegenüber. Ein häufig genanntes Risiko ist die Entstehung von Schatten-IT, wenn Systeme ohne zentrale Steuerung genutzt oder erweitert werden. Zudem stellt die technische Komplexität, insbesondere bei der Integration von Schnittstellen und Datenquellen, eine erhebliche Hürde dar. Auch Akzeptanzprobleme auf Seiten der Nutzer sind nicht zu unterschätzen, da Faktoren wie Innovationsangst oder geringe Selbstwirksamkeit die Nutzung einschränken können[42]. Schließlich erfordern Self-Service-Systeme aufgrund des raschen technologischen Fortschritts eine kontinuierliche Anpassung, um langfristig Nutzen stiften zu können[43].

Insgesamt zeigt sich, dass die identifizierten Erfolgsfaktoren die beschriebenen Herausforderungen unmittelbar adressieren: So kann etwa eine klare Governance die Gefahr von Schatten-IT eindämmen, während ein nutzerorientiertes Design und gezielte Schulungen Akzeptanzprobleme reduzieren.

---

[41] Vgl. Floerecke, *Best-Practices für die Gestaltung von IT-Service-Katalogen und den Einsatz von Self-Service-Portalen*, S. 1514

[42] Vgl. Hartmann/Kerssenfischer/Fritsch/Nguyen, *User Acceptance of Customer Self-Service Portals*, S. 151f

[43] Vgl. Mauludina/Mulyani/Adrianto, *Critical Success Factors for Implementation of Self-Service Business Intelligence in Management Accounting*

2.3 **Large Language Models**

2.3.1 Definition und technische Grundlagen

Large Language Models (LLMs) sind neuronale Netzwerke, die auf der Verarbeitung natürlicher Sprache (Natural Language Processing, NLP) basieren und durch ein selbstüberwachtes Lernverfahren auf riesigen Textkorpora trainiert werden. Sie verfolgen das Ziel, sprachliche Zusammenhänge und Weltwissen zu erfassen, um daraus vielfältige Aufgaben wie Textgenerierung, -klassifikation oder -zusammenfassung zu bewältigen[44].

Die zentrale Idee eines LLMs besteht darin, die Wahrscheinlichkeit von Folgeeinheiten – sogenannten Tokens – in einem gegebenen Text vorherzusagen. Diese Token können Wörter, Wortbestandteile oder Zeichen sein. Das Modell lernt also, welches Token mit welcher Wahrscheinlichkeit auf eine bestimmte Zeichenkette folgt, wodurch ein statistisch fundiertes Sprachverständnis aufgebaut wird[45]. Die Wahrscheinlichkeiten werden dabei anhand eines Wahrscheinlichkeitsverteilungsmodells berechnet, das kontinuierlich optimiert wird, um den Trainingsdaten möglichst genau zu entsprechen[46].

Charakteristisch für LLMs ist ihre enorme Modellgröße. Moderne Sprachmodelle verfügen über Milliarden von Parametern – trainierbare Gewichtungen, die bei der Sprachverarbeitung zum Einsatz kommen. Diese Skalierung ermöglicht es den Modellen, komplexe Sprachmuster und semantische Relationen zu erkennen, die in kleineren Modellen nicht adäquat abgebildet werden können[47].

Im Gegensatz zu klassischen Sprachmodellen, die typischerweise nur unidirektionale Kontexte verarbeiten, basieren viele LLMs auf sogenannten Transformern. Diese Architektur erlaubt es, kontextuelle Informationen sowohl aus vorangegangenen als auch nachfolgenden Tokens gleichzeitig zu berücksichtigen[48].

LLMs markieren einen Paradigmenwechsel in der KI-basierten Sprachverarbeitung, da sie durch Vortrainierung auf allgemeinen Aufgaben universell einsetzbar sind und anschließend für spezifische Anwendungen feinjustiert oder durch gezieltes Prompting gesteuert werden können[49].

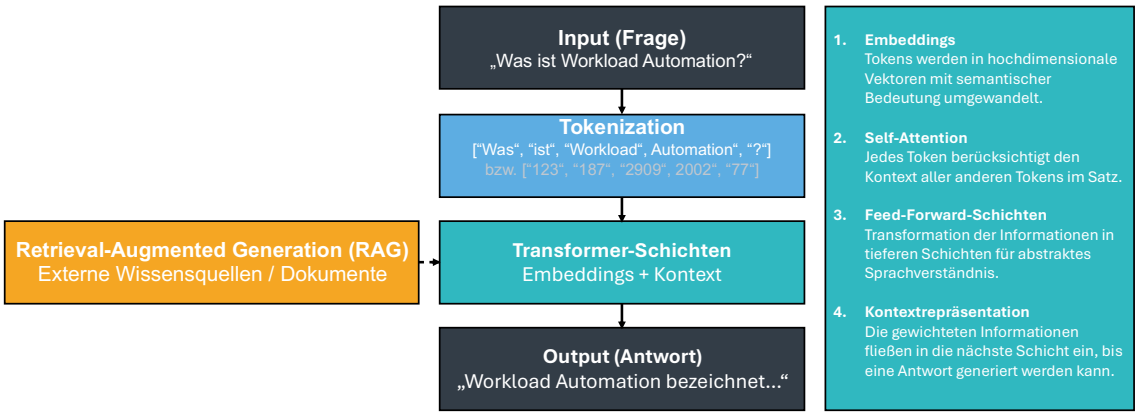
---

[44] Vgl. Xiao/Zhu, *Foundations of Large Language Models*, S. ii
[45] Vgl. Xiao/Zhu, *Foundations of Large Language Models*, S. 6
[46] Vgl. Vaswani et al., *Attention Is All You Need*, S. 3f
[47] Vgl. Devlin et al., *BERT: Pre-training of Deep Bidirectional Transformers*, S. 3
[48] Vgl. Devlin et al., *BERT: Pre-training of Deep Bidirectional Transformers*, S. 1f
[49] Vgl. Xiao/Zhu, *Foundations of Large Language Models*, S. 4

2.3.2 Architektur und Funktionsweise

Large Language Models (LLMs) basieren grundlegend auf der sogenannten Transformer-Architektur, die erstmals von Vaswani et al. vorgestellt wurde. Diese Architektur ersetzt rekurrente und konvolutionale Netzwerke vollständig durch ein Attention-Mechanismus-basiertes System, das sich durch hohe Parallelisierbarkeit und Effektivität bei der Modellierung langer Abhängigkeiten auszeichnet[50]. Der ursprüngliche Transformer besteht aus Encoder und Decoder. Moderne LLMs nutzen jedoch meist entweder nur den Decoder (z. B. GPT) oder nur den Encoder (z. B. BERT). Einem Encoder, der die Eingabesequenz verarbeitet, und einem Decoder, der die Ausgabe erzeugt. Besonders entscheidend ist dabei das Konzept der Self-Attention, mit dem jedes Token der Eingabe auf andere Tokens zugreifen und kontextsensitiv verarbeitet werden kann[51].

Bevor ein LLM überhaupt Texte verarbeiten kann, müssen diese in maschinenlesbare Einheiten, sogenannte Token, zerlegt werden. Die Tokens werden anschließend mithilfe von Embeddings in hochdimensionale Vektoren umgewandelt, die semantische Informationen enthalten[52]. Innerhalb des Modells werden diese Vektoren durch mehrere Schichten der Self-Attention und Feed-Forward-Netzwerke verarbeitet, wobei jedes Token seinen Kontext durch gewichtete Verbindungen zu anderen Tokens innerhalb eines Kontextfensters erhält.



Ein zentraler Aspekt der Funktionsweise von LLMs ist die Art der Trainingsmethoden. Zunächst werden die Modelle im Rahmen des sogenannten Self-supervised Learning auf große Mengen unstrukturierter Textdaten trainiert, etwa durch das Vorhersagen fehlender Tokens (Masked Language Modeling) oder die Fortsetzung eines Textes (Causal Language Modeling)[53]. In späteren Phasen kommt häufig Reinforcement Learning from Human Feedback (RLHF) zum Einsatz, bei dem das Modell anhand menschlicher

[50] Vgl. Vaswani et al., *Attention Is All You Need*, S. 1f
[51] Vgl. Vaswani et al., *Attention Is All You Need*, S. 3f
[52] Vgl. Xiao & Zhu, *Foundations of Large Language Models*, S. 2f
[53] Vgl. Xiao & Zhu, *Foundations of Large Language Models*, S. 7

Präferenzdaten optimiert wird, um hilfreicher und sicherer zu agieren[54]. Ergänzend dazu erlaubt die Technik der Retrieval-Augmented Generation (RAG) die dynamische Anreicherung des Kontextfensters mit externem Wissen, etwa aus Wissensdatenbanken oder Dokumenten, wodurch die Faktengenauigkeit der Modelle gesteigert werden kann[55].

Der Prompt spielt bei der Interaktion mit LLMs eine zentrale Rolle: Nutzer stellen Eingaben in natürlicher Sprache bereit, die das Modell mithilfe seines trainierten Kontexts analysiert. Das Modell verarbeitet dabei nicht nur die aktuellen Eingaben, sondern auch vorherige Token innerhalb eines festgelegten Kontextfensters, um relevante Abhängigkeiten herzustellen[56].

Insgesamt bilden die beschriebenen Komponenten und Prozesse die technologische Grundlage für zahlreiche Anwendungen, die im folgenden Kapitel 2.3.3 im Kontext von Self-Service und Workload-Automatisierung näher betrachtet werden.

2.3.3 Einsatz von LLMs zur Unterstützung von Self-Service und Wissenszugang

Der Einsatz von Large Language Models (LLMs) im unternehmensbezogenen Self-Service hat sich als vielversprechender Weg erwiesen, um den Zugang zu Informationen zu erleichtern und Mitarbeitende im Supportbereich nachhaltig zu entlasten. Besonders im Kundenservice kommen LLM-basierte Chatbots zum Einsatz, die komplexe Benutzeranfragen interpretieren und in natürlicher Sprache beantworten können. Diese Systeme verbessern nicht nur die Antwortqualität, sondern ermöglichen auch eine konsistente Nutzererfahrung rund um die Uhr[57].

Ein weiteres Anwendungsfeld sind semantisch gestützte FAQ-Systeme, bei denen LLMs in der Lage sind, unstrukturierte Nutzerfragen zu analysieren und mit passenden, oft dokumentierten Antworten abzugleichen. Im Unterschied zu traditionellen regelbasierten Systemen erfassen LLMs dabei auch semantische Zusammenhänge, wodurch die Trefferqualität wesentlich verbessert wird[58].

Auch intern unterstützen LLMs zunehmend die intelligente Informationsbereitstellung. Sie dienen als Schnittstelle zu unternehmensweiten Wissensdatenbanken, etwa durch das Generieren von Zusammenfassungen oder das Kontextualisieren relevanter

---

[54] Vgl. Xiao & Zhu, *Foundations of Large Language Models*, S. 172ff
[55] Vgl. Xiao & Zhu, *Foundations of Large Language Models*, S. 77
[56] Vgl. Xiao & Zhu, *Foundations of Large Language Models*, S. 51
[57] Vgl. Dam et al., A Complete Survey on LLM-based AI Chatbots, S. 1f
[58] Vgl. Zhu et al., Large Language Models for Information Retrieval, S. 1f

Passagen aus umfangreichen Dokumenten. Damit tragen sie dazu bei, das implizite Wissen in Organisationen zugänglicher zu machen und interne Supportprozesse zu beschleunigen[59].

Insgesamt zeigt sich, dass LLMs durch ihre dialogische und semantische Kompetenz wertvolle Assistenzsysteme im Self-Service darstellen. Sie erhöhen die Effizienz, reduzieren die kognitive Last von Mitarbeitenden und verbessern den Zugang zu relevantem Wissen.

Im nächsten Abschnitt (2.3.4) wird anschließend aufgezeigt, wie LLMs über die rein interaktive Nutzung hinaus zur technischen Automatisierung und Codierung von Geschäftsprozessen beitragen können.

2.3.4 LLM-gestützte Programmierung und Automatisierung

Der Einsatz von Large Language Models (LLMs) geht über einfache Assistenzfunktionen hinaus und umfasst zunehmend die Unterstützung bei Programmierung und technischer Automatisierung. LLMs können als „Coding Assistants“ fungieren, indem sie Code vervollständigen, Fehler identifizieren und korrigieren sowie Dokumentationen generieren[60]. Besonders relevant für die Automatisierung ist die Fähigkeit von LLMs, Skripte oder API-Verbindungen direkt aus natürlichsprachigen Anforderungen abzuleiten. Damit fungieren sie als Schnittstelle zwischen menschlicher Sprache und maschinenlesbarem Code (z. B. JSON, XML)[61].

Ein besonders innovativer Ansatz ist das Konzept Text2Workflow, das von Minkova et al. (2024) vorgestellt wurde. Dabei werden natürliche Spracheingaben automatisch in ausführbare Workflows im JSON-Format übersetzt. Der Vorteil liegt darin, dass auch komplexe Geschäftsprozesse ohne tiefgehende Programmierkenntnisse formalisiert und automatisiert werden können. Text2Workflow integriert zudem Feedback-Mechanismen, die Nutzern ermöglichen, generierte Abläufe zu überprüfen und anzupassen, wodurch die Zuverlässigkeit erhöht wird[62].

Gerade für die Workload-Automatisierung in Unternehmen spielen strukturierte Austauschformate wie XML oder JSON eine zentrale Rolle. Viele Systeme, von ERP-Software bis hin zu Cloud-Plattformen, setzen auf diese Formate zur Definition von

---

[59] Vgl. Zhu et al., Large Language Models for Information Retrieval, S. 1ff

[60] Vgl. Nijkamp et al., *CODEGEN: An Open Large Language Model for Code with Multi-Turn Program Synthesis, S. 1ff*

[61] Vgl. Weber, *Large Language Models are Pattern Matchers*, S. 2f

[62] Vgl. Minkova et al., *From Words to Workflows: Automating Business Processes, S. 1ff*

Prozessen, Schnittstellen und Datenflüssen. LLMs können hier als Übersetzer agieren: Aus einer einfachen Anweisung wie „Exportiere alle Bestellungen täglich nach XML und übertrage sie an das CRM-System“ wird automatisch eine strukturierte Workflow-Beschreibung in XML oder JSON generiert. Auf diese Weise wird der Schritt von der Geschäftssprache hin zur maschinenlesbaren Prozessdefinition erheblich verkürzt – und damit die Grundlage für Self-Service-Automatisierung gelegt[63].

Darüber hinaus zeigen neuere Studien, dass LLMs auch mit semi-strukturierten und strukturierten Dokumenten wie XML und LaTeX umgehen können. Sie sind in der Lage, solche Daten nicht nur zu verarbeiten, sondern auch zwischen Formaten wie RIS und XML zu konvertieren oder bestehende Strukturen umzustrukturieren[64]. Für Automatisierungsszenarien bedeutet dies, dass LLMs nicht nur Skripte, sondern auch datengetriebene Workflows und Schnittstellendefinitionen generieren können – ein entscheidender Aspekt für Self-Service-Lösungen in der Wirtschaftsinformatik.

Zusammenfassend eröffnen LLMs durch ihre Übersetzungsfähigkeit von natürlicher Sprache zu Code und strukturierten Formaten neue Potenziale für die effiziente Workload-Automatisierung. Allerdings ergeben sich daraus auch Risiken, etwa fehlerhafte Outputs, Sicherheitslücken oder schwer wartbarer Code, die im folgenden Kapitel 2.3.5 näher betrachtet werden.

2.3.5 Grenzen und Herausforderungen

Trotz ihrer Potenziale bergen LLMs erhebliche Risiken bei der Self-Service-Automatisierung und Codegenerierung. Besonders problematisch sind Qualitätsrisiken wie Halluzinationen – also plausibel klingende, aber faktisch falsche Inhalte. Diese treten auch bei strukturierter Ausgabe wie XML oder JSON auf, etwa durch fehlende Elemente oder fehlerhafte Syntax[65].

Ein weiteres Risiko liegt im Bereich Datenschutz und Sicherheit. LLMs können aufgrund unspezifischer Eingaben sensible Informationen ungewollt wiedergeben oder sicherheitskritische Schwachstellen im Code erzeugen[66]. Da viele Self-Service-Nutzer keine tiefgehende technische Prüfung des Codes durchführen, kann dies zu kritischen Fehlern führen.

---

[63] Vgl. Weber, *Large Language Models are Pattern Matchers*, S. 2f
[64] Vgl. Weber, *Large Language Models are Pattern Matchers*, S. 5f
[65] Vgl. Weber, *Large Language Models are Pattern Matchers*, S. 1f
[66] Vgl. Dam et al., *A Complete Survey on LLM-based AI Chatbots*, S. 1ff

Hinzu kommt die eingeschränkte Wartbarkeit und Nachvollziehbarkeit generierten Codes. Automatisch erstellte Lösungen sind oft schlecht dokumentiert, schwer prüfbar und folgen keiner klaren Softwarearchitektur – was langfristig zu Problemen bei der Anpassung und Fehlersuche führt[67].

Ein zentrales Risiko stellt zudem die kognitive Verzerrung dar: LLMs erscheinen oft „überzeugend", auch wenn ihre Aussagen objektiv falsch sind. Nutzer laufen dadurch Gefahr, den Ausgaben zu viel Vertrauen zu schenken[68]. In Kombination mit geringer technischer Expertise kann dies dazu führen, dass fehlerhafter Code unkritisch übernommen wird[69].

Ein vielversprechender Ansatz zur Eindämmung dieser Probleme ist Retrieval-Augmented Generation (RAG), das durch die Anreicherung mit externem Wissen Halluzinationen und Qualitätsprobleme reduzieren kann – näher betrachtet in Kapitel 2.4.

2.4 **Retrieval-Augmented Generation (RAG)**

2.4.1 Konzept und Motivation

Retrieval-Augmented Generation (RAG) ist ein hybrides Verfahren der Künstlichen Intelligenz, das klassische generative Sprachmodelle um externe Wissensquellen ergänzt. Während Large Language Models (LLMs) ihr Wissen ausschließlich aus in Parametern gespeicherten Trainingsdaten schöpfen, erweitert RAG diesen Ansatz durch den dynamischen Zugriff auf zusätzliche Datenbanken oder Dokumentensammlungen. So wird das Sprachmodell durch ein Retrieval-System unterstützt, das relevante Informationen aus externen Quellen beschafft und in den Generierungsprozess einbindet[70].

Im Gegensatz zu reinen LLMs, die bei unbekannten oder aktuellen Fragen zu Halluzinationen oder ungenauen Antworten neigen, ermöglicht RAG eine fundiertere Textgenerierung. Dies gelingt, indem Informationen nicht nur aus dem gelernten Modellwissen stammen, sondern aktiv aus einem durchsuchbaren Wissensspeicher beigesteuert werden[71]. Das Modell agiert somit nicht mehr als alleinige Wissensquelle, sondern als Kontextverarbeiter für extern abgerufene Inhalte.

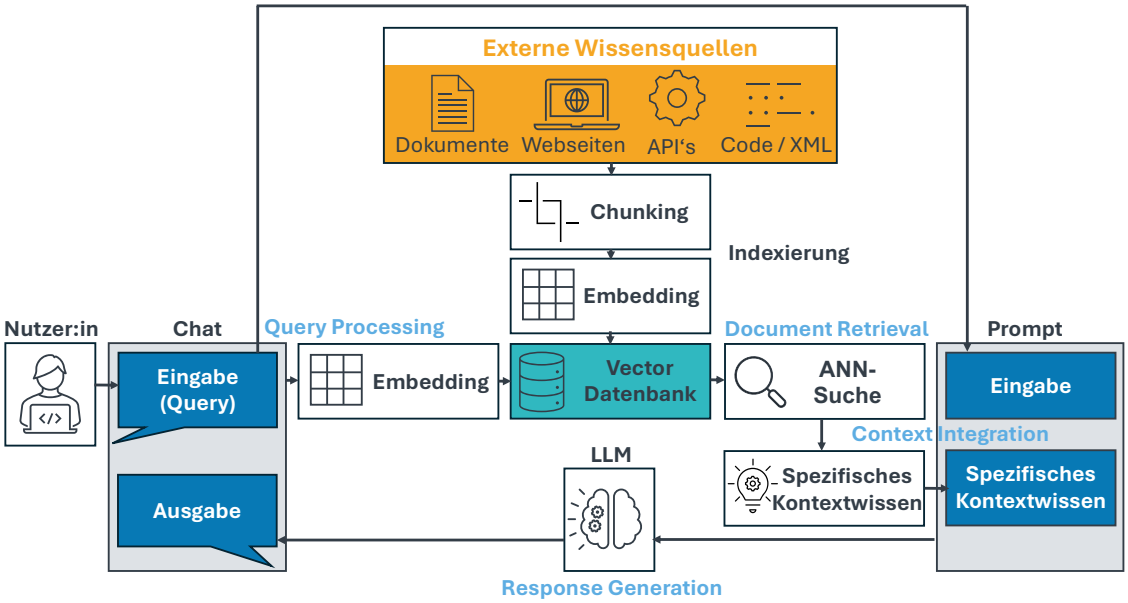
---

[67] Vgl. Dam et al., *A Complete Survey on LLM-based AI Chatbots*, S. 14f
[68] Vgl. Ji et al., *Survey of Hallucination in Natural Language Generation*, S. 4
[69] Vgl. Minkova et al., *From Words to Workflows*, S. 6
[70] Vgl. Honroth, Siebert & Kelbert, Retrieval Augmented Generation (RAG), S. 1f
[71] Vgl. Honroth, Siebert & Kelbert, Retrieval Augmented Generation (RAG), S. 3

Für Self-Service-Anwendungen und die automatisierte Bearbeitung von Workloads bedeutet RAG einen entscheidenden Fortschritt. Es erlaubt Nutzer:innen, komplexe Informationsabfragen auch auf unternehmensinterne Daten zu richten – ohne spezialisiertes Prompt-Design oder manuelle Recherche[72].

2.4.2 Technische Architektur

Die Architektur von Retrieval-Augmented-Generation-Systemen (RAG) folgt einer modularen Pipeline mit vier Phasen: *Query Processing, Document Retrieval, Context Integration* und *Response Generation* (vgl. Abbildung X).



Im **Query Processing** wird die Nutzereingabe in semantische Vektorrepräsentationen überführt, meist durch spezialisierte Embedding-Modelle. **Document Retrieval** erfolgt anschließend über eine Vektor- oder Hybridsuche, die eine externe Wissensbasis – typischerweise Textpassagen in einer Vektordatenbank – nach semantisch ähnlichen Inhalten durchsucht.[73]. Hierbei wird häufig *Maximum Inner Product Search (MIPS)* eingesetzt, unterstützt durch dichte Indexstrukturen[74].

In der Phase der **Context Integration** werden die gefundenen Textfragmente in den Prompt eingebunden. Je nach Architektur wird dabei entweder ein einziges Dokument für die Antwort genutzt (*RAG-Sequence*) oder mehrere Quellen tokenweise integriert (*RAG-Token*).[75]. Diese Flexibilität verbessert insbesondere bei komplexen Aufgaben die Präzision der Antwortgenerierung.

---

[72] Vgl. Honroth, Siebert & Kelbert, Retrieval Augmented Generation (RAG), S. 3
[73] Vgl. Honroth et al., Retrieval Augmented Generation (RAG), S. 4
[74] Vgl. Lewis et al., Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks, S. 3
[75] Vgl. Lewis et al., Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks, S. 3

Die **Response Generation** übernimmt schließlich ein Sprachmodell, etwa BART, das basierend auf dem ursprünglichen Prompt und den eingebetteten Kontextinformationen eine konsistente Antwort erzeugt. Das zentrale Unterscheidungsmerkmal gegenüber klassischen LLMs besteht darin, dass RAG-Systeme auf eine externe, nicht-parametrische Wissensbasis zurückgreifen. Dadurch werden sie nicht nur transparenter, sondern auch leichter aktualisierbar, da neue Daten eingebunden werden können, ohne dass ein aufwändiges Fine-Tuning des Sprachmodells erforderlich ist.[76].

2.4.3 Retrieval-Komponenten und Skalierbarkeit

Die Retrieval-Komponenten bilden das Kernstück von RAG-Systemen, da sie die Verbindung zwischen gespeicherten Wissensbeständen und der generativen Verarbeitung herstellen. Ohne effizientes Retrieval können nachgelagerte Prozesse keine verlässlichen oder aktuellen Informationen liefern, was die Gesamtleistung des Systems unmittelbar einschränkt.

Zentrale Elemente sind **Vektordatenbanken**, die unstrukturierte Daten – etwa Texte, Bilder oder Audiodaten – in hochdimensionalen Vektorrepräsentationen speichern. Dadurch können Informationen nicht nur über exakte Schlüsselwörter, sondern auch über semantische Ähnlichkeiten abgerufen werden[77].

Zur effizienten Suche in diesen Vektorräumen wird häufig die **Approximate Nearest Neighbor (ANN) Search** eingesetzt. Sie reduziert den Rechenaufwand, indem nicht alle Distanzen exakt berechnet, sondern mit hoher Wahrscheinlichkeit die nächsten Nachbarn identifiziert werden. ANN gilt heute als Standardverfahren in vielen KI-Anwendungen und ermöglicht skalierbare semantische Suche. Graphbasierte Strukturen oder Quantisierungsmethoden sichern dabei die Leistungsfähigkeit auch in großskaligen Szenarien[78].

Gerade im Unternehmenskontext ist zudem die **Skalierbarkeit** entscheidend. Mit wachsenden Datenmengen müssen Lösungen niedrige Latenzen und hohe Genauigkeit gewährleisten und zugleich kontinuierlich aktualisierbar sein. Verteilte Architekturen und Cloud-Infrastrukturen erlauben es, große Mengen unstrukturierter Daten in Echtzeit zu verarbeiten und semantisch aufzubereiten, wodurch relevante Informationen konsistent und schnell bereitgestellt werden[79]. Damit bilden leistungsfähige Retrieval-Komponenten

---

[76] Vgl. Honroth et al., Retrieval Augmented Generation (RAG), S. 2f

[77] Vgl. Jing et al., *When Large Language Models Meet Vector Databases: A Survey*, S. 3

[78] Vgl. Dobson et al., *Tutorial on Approximate Nearest Neighbor Search (ANNS) – Techniques and Open Problems*, S. 1f

[79] Vgl. Srivastava et al., *Scaling AI-Driven Solutions for Semantic Search*, S. 1584

eine Grundvoraussetzung für den erfolgreichen Einsatz von RAG in dynamischen Geschäftsumgebungen.

Die in Tabelle 1 dargestellte Gegenüberstellung verdeutlicht, dass RAG für dynamische, informationsintensive Szenarien – wie im Kontext der Workload-Automatisierung – klare Vorteile gegenüber Fine-Tuning aufweist.

2.4.4 Integration von LLMs: RAG vs. Fine-Tuning

Die Integration von Large Language Models (LLMs) in spezifische Anwendungsszenarien kann im Wesentlichen über zwei Ansätze erfolgen: Fine-Tuning und Retrieval-Augmented Generation (RAG). Während beim Fine-Tuning die Modellparameter durch zusätzliche Trainingsphasen an domänenspezifische Daten angepasst werden, erweitert RAG die Basiskompetenzen eines Modells durch die dynamische Einbindung externer Wissensquellen[80].

Fine-Tuning bietet den Vorteil einer hohen Spezialisierung, ist jedoch mit erheblichem Aufwand verbunden. Jede Aktualisierung der Datenbasis erfordert ein erneutes Training, was zu hohen Infrastrukturkosten und längeren Entwicklungszyklen führt. Darüber hinaus besteht das Risiko der Modell-Drift, wenn sich die zugrunde liegenden Daten verändern[81].

RAG hingegen trennt Wissenszugriff (Retriever) und Generierung (Generator). Dadurch können Modelle flexibel mit aktuellen Informationen arbeiten, ohne dass ihre Parameter verändert werden müssen. Dies erleichtert sowohl die Aktualisierbarkeit als auch die Skalierbarkeit solcher Systeme und reduziert die Gefahr, dass veraltete Wissensbestände falsche Ergebnisse erzeugen[82].

Die Vergleichsstudie von Zhang et al. (2023) im Kontext industrieller Code Completion zeigt diese Unterschiede deutlich: Während Fine-Tuning in hochspezialisierten Szenarien eine höhere Präzision liefert, erweist sich RAG durch geringere Kosten, leichtere Wartbarkeit und größere Flexibilität als überlegen[83]. Diese empirischen Befunde bestätigen die theoretisch hervorgehobenen Vorteile von RAG und verdeutlichen dessen Relevanz für dynamische Unternehmenskontexte.

Für Workload-Automatisierung und Self-Service-Portale ist RAG daher besonders geeignet: Endnutzende können mit natürlichsprachlichen Anfragen auf aktuelle

---

[80] Vgl. Gao et al. (2023), *RAG or Fine-tuning?,* S. 2
[81] Vgl. Gao et al. (2023), *RAG or Fine-tuning?,* S. 7
[82] Vgl. Gao et al. (2023), *RAG or Fine-tuning?,* S. 2
[83] Vgl. Gao et al. (2023), *RAG or Fine-tuning?,* S. 5

Konfigurationsdaten, Dokumentationen oder Best Practices zugreifen, ohne dass das Modell kontinuierlich neu trainiert werden muss. Fine-Tuning bleibt hingegen eine Option für stabile, eng umrissene Domänen, in denen sich Daten selten ändern und höchste Präzision erforderlich ist.

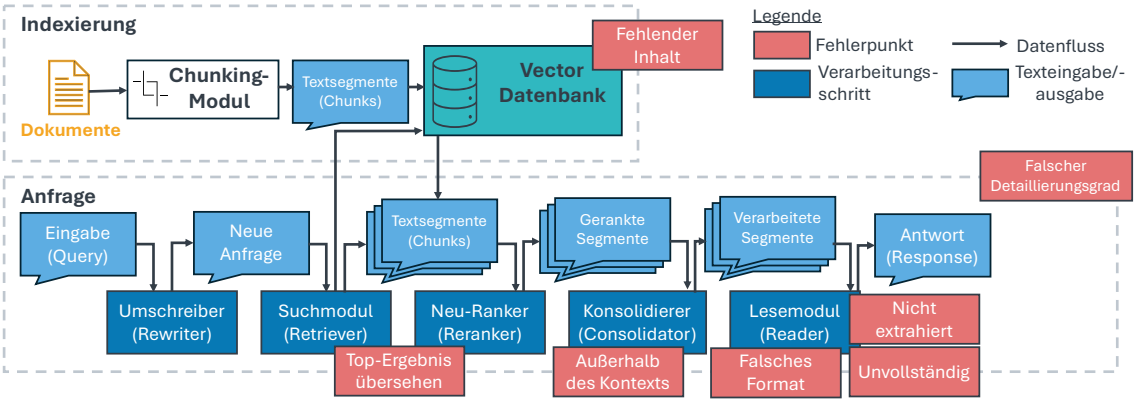
Kriterium	Fehlerpunkt	Erkenntnisse
Prinzip	Anpassung der Modellparameter an spezifische Trainingsdaten	Dynamische Anreicherung durch externes Wissens-Retrieval
Datenaktualität	Veraltet schnell, erfordert erneutes Training	Zugriff auf aktuelle Informationen ohne Retraining
Ressourcenbedarf	Hoher Rechen- und Speicheraufwand, lange Trainingszyklen	Deutlich geringerer Ressourcenbedarf, nur Retriever + Index nötig
Flexibiltät	Stark spezialisiert, aber schwer auf neue Domänen übertragbar	Domänenübergreifend einsetzbar durch Anpassung der Wissensbasis
Qualität	Sehr gute Performance in klar abgegrenzten Domänen	Stärkere Schwankungen durch Retrieval-Qualität
Praxisrelevanz	In der Industrie oft unpraktisch (Kosten, Wartung)	Besser geeignet für dynamische Umgebungen wie Support & Automation

2.4.5 Herausforderungen und Fehlerpotenzial

Obwohl Retrieval-Augmented-Generation (RAG)-Systeme in der Theorie ein vielversprechendes Mittel darstellen, um die Grenzen reiner Sprachmodelle zu überwinden, zeigen sich in der praktischen Umsetzung zahlreiche Herausforderungen. Ein zentrales Problem ist die Halluzinationstendenz von Sprachmodellen: Selbst wenn relevante Dokumente korrekt eingebunden werden, können generierte Antworten sachlich fehlerhaft sein. Dies unterminiert das Vertrauen in RAG-Systeme, insbesondere in wissensintensiven Anwendungsfeldern.

Darüber hinaus ergeben sich **Skalierungsprobleme**. Mit zunehmenden Datenmengen steigen sowohl Latenzzeiten als auch Infrastrukturkosten, da komplexe Vektorsuchen, Re-Ranking-Mechanismen und die Konsolidierung großer Dokumentmengen erhebliche Rechenressourcen erfordern. Hinzu kommt, dass **Evaluation und Monitoring** bislang unzureichend standardisiert sind. Während klassische Retrieval-Metriken wie *Recall* oder *Precision* etabliert sind, fehlen Verfahren, die das Gesamtsystem aus Retrieval und

Generierung ganzheitlich bewerten. Zudem verändern sich Relevanz und Aktualität externer Wissensquellen dynamisch, was eine kontinuierliche Anpassung erfordert[84][85].



Eine praxisorientierte Perspektive bietet die Studie von Barnett et al. (2024), die sieben typische *Fehkerpunkte* identifiziert (vgl. Abbildung 7). Diese decken Schwachstellen entlang der gesamten RAG-Pipeline ab – von fehlenden Inhalten und Rankingproblemen bis hin zu Fehler bei der Kontextintegration. Ergänzend dazu zeigt Tabelle 1 konkrete *Erkenntnisse* die aus empirischen Fallstudien abgeleitet wurden.

Fehlerpunkt	Erkenntnisse
Fehlender Inhalt	Vor dem Aufbau des Index muss geprüft werden, ob die relevanten Inhalte in den Daten überhaupt vorhanden sind.
Falsche Priorisierung	Ranking-Algorithmen müssen feinjustiert werden, da relevante Dokumente sonst übersehen werden.
Außerhalb des Kontextes	Relevante Dokumente entfalten nur dann Nutzen, wenn sie korrekt in den Prompt eingebunden werden.
Nicht extrahiert	Informationen im Prompt müssen so aufbereitet sein, dass das Modell sie zuverlässig extrahieren kann.
Falsches Format	Auch richtige Inhalte sind nutzlos, wenn die Antwort nicht im geforderten Format geliefert wird.
Falscher Detaillierungsgrad	Prompts sollten die gewünschte Detailtiefe klar vorgeben, um Über- oder Untergeneralisierung zu vermeiden.
Unvollständig	Multi-Step- oder Kontrollmechanismen helfen, um alle relevanten Informationen vollständig abzudecken.

[84] Vgl. Yu et al., *Evaluation of Retrieval-Augmented Generation: A Survey*, S. 4
[85] Vgl. Yu et al., *Evaluation of Retrieval-Augmented Generation: A Survey*, S. 3f

Tabelle 1: Tabelle 1: Erkenntnisse zu Fehlerpunkten in RAG-Systemen (eigene Darstellung in Anlehnung an Barnett et al., 2024)

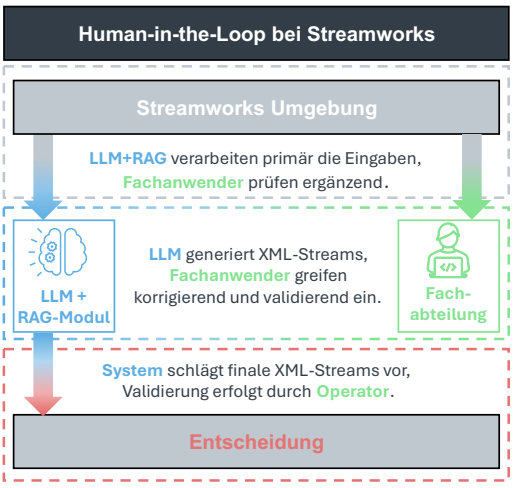
Die technischen Grundlagen von Retrieval-Augmented Generation verdeutlichen die Potenziale, aber auch die Grenzen aktueller Ansätze. Diese führen unmittelbar zu übergeordneten Fragen nach **Vertrauen, Transparenz und Governance**, die für den praktischen Einsatz von KI-gestützter Automatisierung in Unternehmen entscheidend sind und im folgenden Abschnitt betrachtet werden.

2.5 Hybrid-Automatisierung: Mensch-KI-Kollaboration

2.5.1 Human-in-the-Loop und graduelle Automatisierung

Die zunehmende Integration von Künstlicher Intelligenz (KI) in Geschäfts- und IT-Prozesse wirft die Frage auf, wie Automatisierung effizient gestaltet werden kann, ohne die notwendige menschliche Kontrolle zu verlieren. Ein zentraler Ansatz ist das Konzept des **Human-in-the-Loop (HITL)**. Es bezeichnet Systeme, in denen Menschen an kritischen Punkten in automatisierte Abläufe eingreifen können – etwa zur Validierung, Korrektur oder Eskalation. Dies unterscheidet sich von **AI-in-the-Loop-Ansätzen**, bei denen die KI den Hauptakteur darstellt und menschliche Beteiligung lediglich überwachend erfolgt.[86].

Abbildung X zeigt den möglichen Einsatz von Human-in-the-Loop bei Streamworks, bei dem das LLM+RAG-Modul XML-Streams generiert und die Fachabteilung als Operator korrigierend sowie validierend eingreift, bevor das System die finale Entscheidung zur Ausführung trifft.



[86] Vgl. Natarajan et al., *Human-in-the-loop or AI-in-the-loop? Automate or Collaborate?*, S. 1f

Der Vorteil von HITL liegt darin, dass menschliches Urteilsvermögen eingebunden bleibt. Gerade in Situationen, in denen kontextabhängiges Wissen oder ethische Abwägungen erforderlich sind, fungieren Menschen als Kontrollinstanz und reduzieren Risiken wie Verzerrungen oder Fehlentscheidungen[87]. Automatisierung ist dabei kein binärer Zustand, sondern entwickelt sich schrittweise: Viele Organisationen beginnen mit Teilaufgaben, die durch KI unterstützt werden, und steigern den Automatisierungsgrad erst mit wachsendem Vertrauen und verbesserter Systemqualität. Dieses Vorgehen erlaubt Erfahrungen, iteratives Lernen und verhindert, dass kritische Entscheidungen vorschnell allein der Maschine überlassen werden.[88].

Für die **Workload-Automatisierung im Self-Service-Kontext** bedeutet dies eine Balance zwischen Effizienzsteigerung und der Sicherstellung von Qualität und Sicherheit. Studien zeigen, dass gut gestaltete Mensch-KI-Zusammenarbeit die Prozessqualität erhöht, den Service verbessert und die Produktivität steigert[89].

Eng verbunden mit HITL ist das Thema **Vertrauen und Transparenz**. Nutzer akzeptieren KI-gestützte Systeme nur dann, wenn deren Funktionsweise nachvollziehbar bleibt und Verantwortung eindeutig verankert ist[90].

2.5.2 Vertrauen, Transparenz und Governance

Die Akzeptanz KI-gestützter Systeme hängt entscheidend von Vertrauen und Transparenz ab. Anwender müssen nachvollziehen können, wie Ergebnisse zustande kommen, um diese als verlässlich einzustufen. Ein zentrales Konzept hierfür ist Explainable AI (XAI)**,** das Entscheidungsprozesse sichtbar macht und so sowohl Transparenz als auch Verantwortungszuweisung ermöglicht[91].

Darüber hinaus tragen Confidence Scores wesentlich zum Vertrauensaufbau bei. Sie geben an, mit welcher Sicherheit ein Modell eine Entscheidung trifft, und erlauben es Nutzern, Ergebnisse kontextabhängig zu bewerten. Besonders im Self-Service-Umfeld, in dem Endanwender ohne tiefgehende KI-Expertise agieren, sind solche Indikatoren wichtig, um Unsicherheiten abzubauen. [92].

---

[87] Vgl. Kumar & Rongali, *Balancing AI and human collaboration*, S. 564

[88] Vgl. Azevêdo et al., *A path to automated service creation via semi-automation levels*, S. 1f

[89] Vgl. Suram, *Human-AI Symbiosis in Business Process Automation: A Framework for Maximizing Operational Efficiency*, S. 1

[90] Vgl. Gao et al., *HUMAN-CENTERED HUMAN-AI COLLABORATION (HCHAC)*, S. 3

[91] Vgl. Gao et al., *HUMAN-CENTERED HUMAN-AI COLLABORATION (HCHAC)*, S. 3

[92] Vgl. Kumar & Rongali, *Balancing AI and human collaboration*, S. 563

Für Self-Service-Portale ist diese Transparenz besonders wichtig. Hier interagieren Endnutzer direkt mit automatisierten Prozessen, oft ohne vertieftes Fachwissen. Suram zeigt, dass Organisationen, die gut strukturierte Modelle der Mensch-KI-Kollaboration einführen, nicht nur effizientere Arbeitsabläufe erreichen, sondern auch die Servicequalität und das Vertrauen der Nutzer verbessern[93]. Transparenz und klare Kommunikationsmechanismen tragen somit unmittelbar zur Akzeptanz im Self-Service-Bereich bei.

Darüber hinaus ist Governance ein zentraler Faktor für den verantwortungsvollen Einsatz von Automatisierung in Unternehmen. Governance umfasst organisatorische Richtlinien, ethische Standards und Kontrollmechanismen, die den Betrieb sicherstellen. Kumar und Rongali betonen, dass ohne entsprechende Aufsicht KI-Systeme Verzerrungen verstärken, ethische Standards gefährden oder unbeabsichtigte Konsequenzen nach sich ziehen können[94]. Damit Governance wirksam wird, müssen Unternehmen klare Verantwortlichkeiten festlegen und Mitarbeitende gezielt im Umgang mit KI-Systemen schulen.

Das Zusammenspiel von Mensch, KI und Organisation lässt sich dabei als eine Form des human-AI teaming verstehen. Gao et al. beschreiben dies als eine neue Qualität der Zusammenarbeit, die über reine Arbeitsteilung hinausgeht und zu einer evolutionären Weiterentwicklung der Mensch-Maschine-Beziehung führt[95].

Ein nachhaltiger Ansatz für die Zukunft liegt schließlich in hybrider Automatisierung. Azevêdo et al. verdeutlichen, dass ein schrittweises Erhöhen des Automatisierungsgrads zu besserem Verständnis und höherer Reife der Systeme führt[96]. Auf diese Weise entsteht eine Balance zwischen Effizienz und Sicherheit, die für die breite Akzeptanz von KI-gestützter Workload-Automatisierung entscheidend ist.

Die dargestellten theoretischen Konzepte verdeutlichen sowohl die Potenziale als auch die Herausforderungen KI-gestützter Automatisierung. Sie bilden den Bezugsrahmen für die nachfolgende Analyse der Workload-Automatisierung am Beispiel der Plattform Streamworks, die in Kapitel 3 erfolgt.

---

[93] Vgl. Suram, *Human-AI Symbiosis in Business Process Automation: A Framework for Maximizing Operational Efficiency*, S. 1f

[94] Vgl. Kumar & Rongali, *Balancing AI and human collaboration*, S. 565

[95] Vgl. Gao et al., *HUMAN-CENTERED HUMAN-AI COLLABORATION (HCHAC)*, S. 2

[96] Vgl. Azevêdo et al., *A path to automated service creation via semi-automation levels*, S. 1f

3 Analyse der Streamworks-Umgebung

Nach der theoretischen Fundierung wird im Folgenden mit *Streamworks* die in dieser Arbeit untersuchte Workload-Automatisierungsplattform von Arvato Systems vorgestellt. Ziel ist es, das System als Praxisbeispiel in seinen Grundzügen einzuordnen und den Bezug zur Forschungsfrage herzustellen.

3.1 Einführung in Streamworks

Arvato Systems ist ein international tätiger IT-Dienstleister mit Sitz in Deutschland und Tochterunternehmen der Bertelsmann SE. Mit über 40 Jahren Erfahrung im Rechenzentrumsbetrieb entwickelt Arvato Systems spezialisierte IT-Lösungen, darunter die Plattform *Streamworks* zur Workload-Automatisierung[97]. Die Software wurde ursprünglich für den Eigenbetrieb konzipiert und wird inzwischen auch externen Kunden zur Verfügung gestellt.

Streamworks fungiert als zentrale Steuerungsinstanz für IT-gestützte Prozesse, indem es Abläufe über verschiedenste Plattformen hinweg automatisiert, überwacht und optimiert. Die Lösung adressiert dabei komplexe Anforderungen in heterogenen IT-Landschaften und unterstützt eine Vielzahl von Betriebssystemen, Applikationen und Cloud-Diensten.

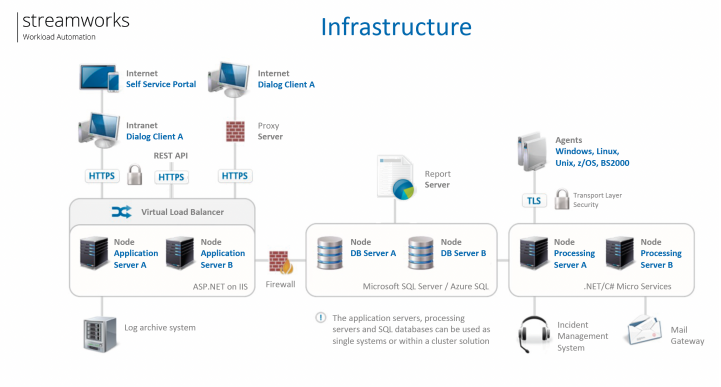
Aufbauend auf dieser Einführung wird im nächsten Abschnitt die technische Architektur von Streamworks beschrieben, da sie die Grundlage für das Verständnis des Stream-Erstellungsprozesses bildet.

3.2 Technische Architektur & XML (~1,5 S. / 375 W)

Um die Funktionsweise von Streamworks nachvollziehen zu können, ist zunächst die technische Architektur zu betrachten. Diese umfasst zentrale Systemkomponenten, Schnittstellen sowie die XML-Strukturen, die das Fundament der Prozessmodellierung bilden.

---

[97] Vgl. https://www.arvato-systems.de/infrastruktur-betrieb/cloud-data-center-services/workload-automation-streamworks



Die technische Architektur von Streamworks basiert auf einer modularen Systemstruktur, die zentrale und dezentrale Komponenten miteinander verbindet. Im Kern steht eine Microsoft SQL Server-Datenbank, in der Stammdaten, Laufzeitprozesse und Systemmeldungen gespeichert werden. Der Processing Server übernimmt die zentrale Steuerung der Abläufe, kommuniziert mit den Agenten und sorgt für die Ausführung der definierten Jobs. Ergänzend stellt der Application Server (auf Basis von Microsoft IIS) moderne Schnittstellen bereit, über die sowohl ein Desktop-Client für Administratoren als auch ein webbasiertes Self-Service-Portal für Fachanwender angebunden sind[98].

Streamworks unterstützt eine Vielzahl von Plattformen, darunter Windows, Unix/Linux sowie Mainframe-Systeme wie z/OS oder BS2000[99]. Die eingesetzte Agentensoftware ist leichtgewichtig, ressourcenschonend und ermöglicht die Integration heterogener Zielsysteme. Für die Einbettung in bestehende IT-Landschaften stehen verschiedene Schnittstellen zur Verfügung, darunter eine REST-API zur systemübergreifenden Kommunikation, ein Command Line Interface (CLI) für Automatisierungsskripte sowie Export- und Import-Funktionen zur Übertragung von Konfigurationen im XML-Format[100].

Eine besondere Rolle nimmt das XML-basierte Datenmodell ein. Sämtliche Streams und Jobs werden in strukturierter XML-Form definiert, was eine einheitliche Darstellung sowie Wiederverwendbarkeit und Austauschbarkeit von Abläufen sicherstellt.[101]

Für die vorliegende Arbeit ist die XML-Nutzung auch im Hinblick auf KI relevant. Die klar strukturierte Repräsentation ermöglicht eine automatische Generierung und Validierung durch Large Language Models (LLMs), wie sie in den Kapiteln 4 und 5 behandelt wird. XML bildet damit die Grundlage, um vollständige Streamdefinitionen automatisiert zu

---

[98] Vgl. Streamworks Produktbeschreibung, S. 3f
[99] Vgl. Streamworks Produktbeschreibung, S. 6
[100] Vgl. Streamworks Produktbeschreibung, S. 9
[101] Vgl. Streamworks Produktbeschreibung, S. 9ff

erzeugen, zu prüfen und anzupassen – ein entscheidender Schritt in Richtung intelligenter Self-Service-Automatisierung.

3.3 Der aktuelle Stream-Erstellungsprozess (Ist-Analyse)

Im Folgenden wird der aktuelle Prozess zur Erstellung von Streams bei Arvato Systems untersucht. Die Darstellung basiert auf Experteninterviews und internen Dokumentationen und ermöglicht eine praxisnahe Abbildung des Status quo.

Die Erstellung neuer Streams erfolgt aktuell in einem stark manuellen, von Kommunikation geprägten Ablauf. Anforderungen erreichen das zuständige Orchestration-Team über unterschiedliche Eingangskanäle: Neben strukturierten Ticket-Systemen oder Helpdesk-Formularen werden vielfach auch E-Mails, Excel- oder Word-Dokumente genutzt. Während standardisierte Vorlagen eine gewisse Revisionssicherheit gewährleisten, dominieren in der Praxis heterogene Formate und unvollständige Angaben, die den Prozess erheblich verlangsamen[102].

Zur Bearbeitung werden bestimmte Pflichtinformationen benötigt, wie etwa der Streamname nach Konvention, Programme und Varianten für SAP-Jobs, Intervalle und Startzeiten, Abhängigkeiten, Benutzerrechte sowie Ansprechpartner. Werden diese Daten nicht in vollständiger und konsistenter Form geliefert, entstehen Rückfragen und Abstimmungsschleifen. Besonders problematisch ist dies bei SAP-Jobs, da hier fehlerhafte oder veraltete Serverangaben die Umsetzung vollständig blockieren können[103].

Der Prozess gliedert sich in mehrere Schritte: (1) Eingang und Sichtung der Anforderung, (2) Prüfung auf Vollständigkeit, (3) Klärung fehlender Informationen, (4) manuelle Umsetzung im System durch XML-Stream-Erstellung, (5) Tests in der QS-Umgebung und (6) Freigabe für die Produktion. Kleinere Anpassungen können in wenigen Minuten erledigt werden, während die Umsetzung neuer Streams im Durchschnitt ein bis zwei Wochen beansprucht. In komplexen Fällen verlängern sich die Durchlaufzeiten aufgrund von Terminabstimmungen oder ausstehenden Rückmeldungen auf bis zu fünf Wochen[104].

---

[102] Vgl. Zusammenfassung der Experteninterviews, S. 3
[103] Vgl. Zusammenfassung der Experteninterviews, S. 4
[104] Vgl. Zusammenfassung der Experteninterviews, S. 4f

Die größten Schwierigkeiten ergeben sich aus unvollständigen Pflichtangaben, widersprüchlichen Vorlagen, sprachlichen Hürden und einem fehlenden Verständnis für Abhängigkeiten seitens der Fachbereiche. Das Orchestration-Team sieht dadurch einen hohen Abstimmungsbedarf, der den Prozess stark verzögert und zu zusätzlicher Fehleranfälligkeit führt[105].

Aus dieser Analyse ergeben sich zentrale operative Herausforderungen, die im nächsten Kapitel systematisch aufgearbeitet und im Hinblick auf ihre Auswirkungen bewertet werden.

3.4 **Herausforderungen (~1 S. / 250 W)**

Die Untersuchung des Ist-Prozesses verdeutlicht verschiedene Problemfelder, die sowohl Effizienz als auch Qualität beeinträchtigen. Im Folgenden werden diese Herausforderungen strukturiert dargestellt.

Ein zentrales Problem liegt in der **hohen Heterogenität der Kundenanforderungen**. Unterschiedliche Eingangskanäle und Formate – von frei formulierten E-Mails bis zu Excel-Vorlagen – führen zu Inkonsistenzen und erschweren eine standardisierte Weiterverarbeitung.

Eng damit verbunden ist die **mangelnde Standardisierung**. Pflichtangaben werden häufig unvollständig oder fehlerhaft geliefert, sodass das Orchestration-Team wiederholt Rückfragen stellen muss. Dies erzeugt nicht nur Mehraufwand, sondern verlängert die Durchlaufzeiten erheblich[106].

Ein weiterer Aspekt ist das **Fehlen automatisierter Vorprüfungen**. Derzeit existieren keine systematischen Mechanismen, um die Vollständigkeit und Validität von Kundendaten beim Eingang automatisch zu überprüfen. Dadurch verschiebt sich die Verantwortung für die Datenqualität vollständig auf das Orchestration-Team, was zu ineffizienten Schleifen führt.

Besonders deutlich wird die Problematik am Beispiel von **SAP-Jobs**: Diese wären aufgrund klarer Pflichtparameter prinzipiell gut standardisierbar. In der Praxis scheitert eine automatisierte Umsetzung jedoch häufig an unvollständigen oder falschen Eingaben der

[105] Vgl. Zusammenfassung der Experteninterviews, S. 5
[106] Vgl. Zusammenfassung der Experteninterviews, S. 5f

Fachbereiche. Damit bleibt auch bei standardisierbaren Szenarien ein hoher manueller Aufwand bestehen[107].

Zusammenfassend lässt sich festhalten, dass die Herausforderungen im Betrieb vor allem aus **Inkonsistenz, fehlender Standardisierung und mangelnden Automatisierungsansätzen in der Anforderungsaufnahme** resultieren. Diese Faktoren führen zu hohen Kommunikationskosten, langen Durchlaufzeiten und einer geringen Skalierbarkeit des Prozesses – und verdeutlichen damit die Notwendigkeit grundlegender Verbesserungen in Richtung Self-Service und KI-gestützter Automatisierung.

Trotz dieser Einschränkungen lassen sich deutliche Potenziale für Standardisierung und Automatisierung erkennen, die in Kapitel 3.5 näher betrachtet werden.

3.5 Automatisierungspotenziale & Self-Service

Die Analyse der bestehenden Stream-Erstellungsprozesse und der damit verbundenen Herausforderungen verdeutlicht, dass insbesondere strukturierte und regelmäßig wiederkehrende Anforderungen ein hohes Automatisierungspotenzial aufweisen. Dies betrifft vor allem einfache Standardjobs – wie beispielsweise SAP-Exporte – bei denen nur wenige, eindeutig definierbare Parameter benötigt werden[108]. In solchen Fällen kann ein Self-Service-Portal eine erhebliche Effizienzsteigerung ermöglichen, wenn Nutzer relevante Informationen wie Programmname, Job-Variante und Zeitsteuerung selbst eingeben können.

Einige Fachpersonen betonen, dass vorbereitende Tätigkeiten wie das Einlesen von standardisierten Excel-Vorlagen technisch bereits automatisierbar sind, sofern sich die Nutzer an feste Strukturen halten. Die Validierung der Eingabedaten anhand solcher Templates könnte durch automatisierte Prüfprozesse erfolgen, etwa mithilfe von Pflichtfeldkontrollen und Namenskonventionen. Ergänzend kann ein Self-Service-System Rückfragen weitgehend vermeiden, indem es technische Plausibilitätsprüfungen direkt integriert[109].

Die Vorteile einer solchen Lösung liegen nicht nur in der Zeitersparnis. Ein Experte hob hervor, dass ein Self-Service-Portal die Verständigung zwischen operativen und fachlichen Einheiten vereinheitlichen könne. Insbesondere bei mehrsprachigen Teams kann

---

[107] Vgl. Zusammenfassung der Experteninterviews, S. 6
[108] Vgl. Zusammenfassung der Experteninterviews, S. 4
[109] Vgl. Zusammenfassung der Experteninterviews, S. 5

dies zu einer signifikanten Reduktion von Missverständnissen führen. Ein weiterer Mehrwert liegt im Rückgang wiederholter Rückfragen: Durch gezielte Felder und strukturierte Eingabeformulare wird die Zahl der Abstimmungsschleifen erheblich reduziert[110].

Zugleich zeigen die Interviews die Grenzen solcher Automatisierungsansätze auf. Bei individuellen Sonderfällen mit komplexer Logik oder domänenspezifischen Anforderungen bleibt menschliche Expertise unverzichtbar. Ein Fachvertreter betonte, dass automatisierte Systeme nicht in der Lage sind, semantische Mehrdeutigkeiten oder implizite Anforderungen korrekt zu erfassen. In diesen Fällen ist eine Einbindung menschlicher Prüfprozesse erforderlich – im Sinne des in Kapitel 2.5 beschriebenen Human-in-the-Loop-Prinzips.

Darüber hinaus muss sichergestellt sein, dass automatisierte Prozesse nicht auf Kosten von Qualität und Nachvollziehbarkeit gehen. Mehrere Stimmen aus der Praxis betonten, dass eine robuste Governance-Struktur, inklusive Rollen- und Rechtekonzept, unerlässlich sei. Auch bei standardisierten Prozessen sollten Stichprobenprüfungen und Protokollierung etabliert sein, um langfristig Vertrauen in das System zu gewährleisten.

Insgesamt lässt sich feststellen: Die automatisierte Erstellung von Workloads im Self-Service-Ansatz ist bei standardisierten, wiederkehrenden Anwendungsfällen realistisch und effizient umsetzbar. Erfolgsfaktoren sind dabei: die Qualität der Eingangsdaten, verbindliche Eingaberegeln, Validierungsmechanismen sowie klare Abgrenzung zwischen Automatisierung und menschlicher Kontrolle bei komplexen Sonderfällen.

3.6 Abgeleitete Anforderungen

Aus der Analyse ergeben sich zentrale Anforderungen für das im folgenden Kapitel entwickelte Konzept:

1. **Zentrale, einheitliche Eingabemaske:** Statt uneinheitlicher E-Mails oder Excel-Dateien müssen alle Anfragen über ein strukturiertes Self-Service-Portal erfolgen, um Medienbrüche und Fehlerquellen zu vermeiden.
2. **Verpflichtende Eingabefelder mit Validierung:** Durch technische Prüfmechanismen wie XSD-Schema-Validierung sollen nur korrekte und vollständige Anforderungen übermittelt werden können.
3. **Klare Rollen- und Berechtigungskonzepte:** Die Nutzung des Portals erfordert ein Governance-Framework, das definiert, welche Nutzer welche Streams erstellen oder freigeben dürfen.

[110] Vgl. Zusammenfassung der Experteninterviews, S. 3f

4. **HITL-Prinzip für Sonderfälle:** Für komplexe, potenziell fehleranfällige Konfigurationen muss eine gezielte menschliche Review-Komponente vorgesehen sein.
5. **Nachvollziehbarkeit und Dokumentation:** Alle automatisch erzeugten Streams müssen versioniert, geloggt und prüfbar dokumentiert sein, um Transparenz und Qualitätssicherung zu gewährleisten.

Diese Anforderungen bilden die konzeptionelle Grundlage für die Architektur und Funktionalität des in Kapitel 4 beschriebenen Self-Service-Systems.

4 Konzeption und Design (~6–7 Seiten / ~1.600–1.750 Wörter)

4.1 Zielarchitektur (~1,5 S.)

Übersicht Komponenten: Portal, LLM, RAG, XSD-Validator

4.2 Self-Service-Portal (~1,5 S.)

- UI-Konzept: Pflichtfelder, Eingabevalidierung
- Rollen- und Rechtekonzept

4.3 XML-Generierung

- Workflow: Input → LLM → XML → XSD-Validierung
- Logging & Nachvollziehbarke

4.4 RAG-Support-System

- Q&A-Architektur
- Datenbasis: Handbücher, Dokumentation
- Integration in Portal

5 Implementierung (~8–9 Seiten / ~2.000–2.250 Wörter)

5.1 Frontend

- Frameworkwahl (React/Vue)
- UI-Komponenten & Formulare

5.2 LLM-Integration

- Schnittstelle zu Modell (z. B. OpenAI)
- Prompt-Engineering für XML
- Rückgabeverarbeitung

5.3 XSD-Validierung

- Aufbau Schema
- Automatisierte Prüfungen & Fehlermeldungen

5.4 RAG-Integration

- Vektorbasierte Suche
- LLM-Kopplung
- UI-Integration

6 Evaluierung und Testing (~6–7 Seiten / ~1.600–1.750 Wörter)

6.1 Testmethodologie

Testarten: Unit, Integration, Fachanwender-Tests

6.2 Qualitätsbewertung generierter Streamworks-Konfigurationen

- Korrektheit & Vollständigkeit
- Vergleich zu manuellen XMLs

6.3 Effizienzsteigerung durch Automatisierung

- Vergleich manuell vs. Self-Service + KI
- Kennzahlen: Zeit, Fehlerquote

6.4 RAG-Bewertung

- Qualität Antworten
- Nutzerfeedback

7 Ergebnisse und Bewertung (~3–4 Seiten / ~1.000 Wörter)

- Zusammenfassung aller Ergebnisse
- Bewertung Zielerreichung
- Nutzen für Unternehmen

7.1 Funktionalität und Korrektheit der XML-Generierung

7.2 Performance und Skalierbarkeit des Self-Service-Systems

7.3 Benutzerfreundlichkeit und Adoptionsrate

7.4 Mehrwert des integrierten Q&A-Support-Systems

8 Kritische Reflexion (~3 Seiten / ~750 Wörter)

- Grenzen der LLM-Generierung
- Herausforderungen der Validierung
- Skalierbarkeit & Integration

8.1 Limitationen der LLM-basierten XML-Generierung

8.2 Herausforderungen bei der Validierung und Qualitätssicherung

8.3 Skalierbarkeit und Integration in bestehende Workflows

9 Fazit und Ausblick (~3 Seiten / ~750 Wörter)

- Kernergebnisse
- Beitrag der Arbeit
- Zukünftige Entwicklungen

9.1 Zusammenfassung der Ergebnisse

9.2 Beitrag zur Streamworks-Workload-Automatisierung

9.3 Zukünftige Entwicklungsmöglichkeiten

10 Literaturverzeichnis

Geck, R.-L. (2025): **Zusammenfassung der Experteninterviews zur Streamworks-Automatisierung**. Unveröffentlichtes Dokument im Rahmen der Bachelorarbeit an der FHDW, Paderborn

11 Anhang

11.1 Zusätzliche Abbildungen und Tabellen

11.2 Codebeispiele