FH AACHEN
UNIVERSITY OF APPLIED SCIENCES

Fachochschule Aachen, Campus Jülich

Fachbereich 09 - Medizintechnik und Technomathematik
Studiengang Angewandte Mathematik und Informatik

Seminararbeit

Entwicklung eines Konzepts und Prototyps zur Interaktion mit Prozessmodellen mittels eines Model Context Protocol-unterstützten Chatbots

Autor:
Fidel Adrian Botello Luna, 3624179

Betreuer:
Prof. Dr. rer. nat. Alexander Voß
Tim Hommen M. Sc.

Aachen, 15. Dezember 2025

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit eigenständig angefertigt und ausschließlich die im Literaturverzeichnis vermerkten Quellen verwendet habe.

Im Rahmen der Erstellung dieser Arbeit wurde das KI-System “KI.connect.nrwünterstützend zur sprachlichen Überarbeitung sowie zur fachlichen Reflexion und Präzisierung eigenständig entwickelter Argumente genutzt. Eine Übernahme von KI-generierten Textenoder inhaltlichen Lösungsvorschlägen erfolgte nicht. Sämtliche fachlichen Aussagen, Bewertungen und Schlussfolgerungen wurden eigenständig erarbeitet und verantwortet. Die Nutzung erfolgte im Einklang mit der Zweckbestimmung des Systems sowie unter Beachtungdatenschutz- und urheberrechtlicher Vorgaben.

Name: Fidel Adrian Botello Luna
Aachen, 15. Dezember 2025

Zusammenfassung

Die Abteilung Produktionsmanagement des Werkzeugmaschinenlabors (WZL) der RWTH Aachen nutzt das Webtool *Process Flow* zur Modellierung und Analyse von Produktionsprozessen. Um die Interaktion mit diesen Prozessmodellen für Nutzer ohne tiefergehende Expertise über das Prozessmodell sowie Prozessmodellierung im Allgemeinen zu vereinfachen, entwickelt diese Arbeit ein Konzept und einen Prototyp für einen Chatbot, der durch das *Model Context Protocol (MCP)* unterstützt wird und eine neue Interaktionschnittstelle für den Nutzer bietet.

Der entwickelte Ansatz integriert ein Large Language Model (Llama 3.1 8B) über eine MCP-basierte Architektur in die bestehende Process Flow-Umgebung. Dies ermöglicht es Anwendern, Prozessmodelle über natürliche Spracheingaben zu befragen und zu bearbeiten, ohne direkt mit der zugrundeliegenden Modellierungssyntax interagieren zu müssen. Die Architektur gewährleistet dabei eine sichere und konsistente Anbindung an die Prozessdaten über das Process Flow-Backend.

Die Arbeit umfasst die Spezifikation der Anforderungen, die konzeptionelle Gestaltung der Systemkomponenten, inklusive MCP-Client, MCP-Server und LLM-Integration, sowie die prototypische Umsetzung eines minimal funktionsfähigen Systems. Die Evaluation mittels repräsentativer Testfälle validiert die grundlegende Funktionalität des Prototyps und zeigt, dass das Gesamtkonzept eine zuverlässige Schnittstelle für Prozessmodellierung darstellen kann.

Das Ergebnis ist ein erweiterbares Konzept und ein lauffähiger Prototyp, der die Grundlage für eine natürliche, KI-gestützte Interaktion mit Prozessmodellen im Kontext der betrieblichen Prozessanalyse bildet.

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Die strukturierte Erfassung und Analyse von Geschäftsprozessen stellt eine grundlegende Voraussetzung für die Wettbewerbsfähigkeit moderner Unternehmen dar. Sie enthält ein hohes Potenzial zur maßgeblichen Verbesserung von Produktions- wie auch Geschäftsprozessen im Allgemeinen. Konkret können durch eine systematische Prozessoptimierung zentrale Faktoren wie die Durchlaufzeit reduziert, der Aufwand von Mitarbeitenden minimiert und die Fehleranfälligkeit von Arbeitsabläufen verringert werden.
Die Prozessmodellierung bildet hierfür die entscheidende Grundlage. Sie schafft eine visuelle und formale Repräsentation von Arbeitsabläufen, die es ermöglicht, diese zu verstehen und detailliert zu analysieren. Dabei wird zwischen Ist-Prozessen, der reale Prozess, und Soll-Prozessen, die den Zielzustand darstellen, unterschieden. Dieser Vergleich ermöglicht es, gezielt Abweichungen und damit Optimierungspotenziale zu identifizieren und dient der langfristigen Sicherung der Prozesstreue. Darüber hinaus dient ein Prozessmodell als Instrument zur Wissensvermittlung und zum Onboarding neuer Mitarbeiter.
Vor diesem fachlichen Hintergrund befasst sich die Abteilung Produktionsmanagement des Werkzeugmaschinenlabors (WZL) der RWTH Aachen mit der Modellierung und Analyse industrieller Produktionsprozesse. Das Ziel ist es, Unternehmen dabei zu unterstützen, ihre Abläufe effizienter, transparenter und zukunftssicher zu gestalten.
Um diese Aufgabe methodisch zu unterstützen, wurde in der Abteilung die Prozessmodellierungssprache Aixperanto wie auch das darauf basierende Tool Process Flow entwickelt. Dieses Werkzeug dient dazu, Produktionsprozesse mithilfe von Swimlane-Diagrammen strukturiert abzubilden. Die vorliegende Arbeit knüpft an diese Entwicklung an und adressiert die Herausforderung, den Zugang zu den in Process Flow enthaltenen Prozessmodellen weiter zu vereinfachen und ihre praktische Nutzbarkeit zu erhöhen.

1.2 Handlungsbedarf

Trotz ihres hohen Nutzens sind bestehende Methoden zur Erfassung und Interaktion mit Prozessmodellen oft mit erheblichen Hürden verbunden. Die manuelle Erstellung und Pflege umfangreicher Modelle erfordert einen hohen zeitlichen Aufwand sowie spezifische Expertise in der Prozessmodellierung wie auch über dem spezifischen Prozessmodell und internes Wissen

des Unternehmens. Selbst für erfahrene Anwender kann die Navigation in komplexen grafischen Darstellungen oder tabellarischen Übersichten mühsam sein.
Als repräsentatives Beispiel dient das Prozessmodell des Kundenauftragszyklus. Dieses Modell bildet einen typischen Geschäftsablauf von der Kundenanfrage bis zur Produktauslieferung ab und umfasst insgesamt zehn wesentlichen Prozessschritte, Entscheidungspunkte und beteiligte Rollen eines realen Geschäftsprozesses, wie sie auch später in Kapitel 2.1 erläutert werden. In der praktischen Anwendung sind derartige Prozessmodelle häufig um ein Vielfaches umfangreicher und komplexer strukturiert, was die Herausforderungen der Prozessmodellierung unterstreicht. Die regelmäßige Aktualisierung und praktische Nutzung eines solchen Modells ist ohne vereinfachten Zugang deutlich erschwert. Diese Umstände führen in der Praxis häufig dazu, dass Prozessmodellierung nur unregelmäßig oder in reduzierter Form durchgeführt wird. In der Folge verlieren die Modelle an Aktualität und Aussagekraft, was ihre Vertrauenswürdigkeit und ihren praktischen Nutzen mindert.
Es besteht daher ein klarer Bedarf an einer neuen Form der Interaktion mit Prozessmodellen. Eine solche Lösung muss den Aufwand und die erforderliche Expertise für die Nutzung deutlich senken, um die regelmäßige Pflege und Nutzung von Prozesswissen zu fördern. Ein vielversprechender Ansatz ist die Entwicklung eines Chatbots, welcher es ermöglicht, über natürliche Sprache mit Prozessmodellen zu interagieren. Dies würde die Hürden in der Prozessmodellierung deutlich verringern und eine regelmäßigere Nutzung sowie Pflege der Modelle fördern.

1.3 Ziel

Das übergeordnete Ziel dieser Arbeit ist die Konzeption und prototypische Implementierung eines Chatbot Systems, das eine effiziente und nutzerfreundliche Interaktion mit Prozessmodellen ermöglicht. Als durchgängiges Beispiel dient dabei der bereits erwähnte Kundenauftragszyklus, anhand dessen die entwickelten Funktionen evaluiert werden. Das System soll es Nutzenden erlauben, in natürlicher Sprache Fragen zum Prozessmodell zu stellen, Erläuterungen zu erhalten und gezielte Änderungen am Modell vorzunehmen.
Durch die Nutzung des **Model Context Protocols (MCP)** als Integrationsschicht soll sichergestellt werden, dass der Chatbot konsistent auf die aktuellen Prozessdaten zugreifen und zuverlässige Antworten generieren kann. Durch die Senkung der Anforderung an den Nutzer soll diese Arbeit dazu beitragen, dass Prozessmodelle langfristig aktuell, verständlich und nutzbar bleiben und so ihren vollen Wert für Analyse, Optimierung und Wissenssicherung entfalten können.

2 Grundlagen und Herausforderungen

Bei diesem Kapitel handelt es sich einserseits um eine Darstellung der Grundlagen welche für das Verständnis der übrigen Arbeit erforderlich sind. Dazu gehört eine Einführung in die Prozessmodellierung, wie auch Large Language Models. Andererseits stellt es auch die aktuellen Herausforderungen in der Praxis dar, die sich bei der Prozessmodellierung und der Anbindung von KI-Systemen ergeben.

2.1 Prozessmodellierung

Unter Prozessmodellierung versteht man die systematische Beschreibung, Analyse und Darstellung von Prozessen bzw. Geschäftsprozessen. Ziel ist es, Abläufe nachvollziehbar und standardisiert zu gestalten, sodass sie verstandenden, bewertet und verbessert werden können. Prozessmodelle dienen damit sowohl als Kommunikationsmittel zwischen Stakeholdern als auch als Grundlage für das Verständnis, wie auch die Optimierung von Prozessen. [1]
Prozessmodelle bestehen aus grundlegenden Elementen, welche unabhängig von der verwendeten Modellierungsprache ähnlich strukturiert sind. Dazu gehören **Aktivitäten**, welche spezifische Aufgaben im Rahmen des Prozesses entsprechen, sowie **Ereignisse**, welche den Prozess initiieren, abschließen oder fortführen z. B. in Form einer eingetroffenen Nachricht oder das Erreichen einer Frist. Dabei dienen **Handlungspfade/Sequenzflüsse** dazu Aktivitäten und Ereignisse mit einander zu verbinden. Zusätzlich werden häufig **Bedingungen/Gateways** dargestellt, die an bestimmten Stellen alternative **Handlungspfade/Sequenzflüsse** ermöglichen. Auch Daten, Ressourcen oder beteiligte Rollen lassen sich in vielen Modellierungsprachen einbinden und verdeutlichen, um bsw. mit Hilfe von **Swimlanes** Auskunft geben zu können wer für die Durchführung bestimmter Aktivitäten verantwortlich ist. Die grafische Darstellung dieser Elemente hilft dabei, die Logik eines Prozesses klar und eindeutig zu vermitteln, selbst wenn der zugrunde liegende Ablauf eine hohe Komplexität aufweist. [1, 2]

Diese Struktur wird im Beispiel des Kundenauftragszyklus deutlich. Aktivitäten wie „Anfrage eingeben“ oder „Anfrage prüfen“stellen klare Aufgaben in dem Prozessmodell dar, Ereignisse wie „neue Anfrage erhalten“ und die Bedingung „Anfrage valide“ bilden den gesamten Prozess ab. Swimlanes visualisieren dabei die Verantwortlichkeiten verschiedener Abteilungen. Im Fall des Kundenauftragszyklus sind dies der Innendienst, Vertrieb und die Entwicklungsabteilung.
Für die Beschreibung von Prozessen stehen verschiedene Modellierungssprachen und Notationen zur Verfügung, die sich jeweils in ihrem Abstraktionsgrad, ihrer Zielgruppe und ihrem Anwendungsfokus unterscheiden. Die **Business Process Model and Notation (BPMN)** hat sich heute als einer der am weitesten verbreiteten Standards etabliert. Sie zeichnet sich durch einen umfangreichen, aber standardisierten Elementen aus und eignet sich besonders für die modellübergreifende Darstellung von Geschäftsprozessen. Ein weiterer etablierter Ansatz sind die **Ereignisgesteuerte Prozessketten (EPK)**. Sie verwenden eine geringe Anzahl an Elementen, die im Vergleich zu BPMN weniger komplexe Modellierungen ermöglichen. Dieser Ansatz führt zu einer kompakten Darstellung und wird insbesondere im deutschsprachigen Raum eingesetzt. **UML-Aktivitätsdiagramme** stammen aus der Softwareentwicklung und eignen sich für die detaillierte Modellierung von Aktivitäten innerhalb eines Systems, insbesondere in IT-nahen Kontexten. [1, 2]

2.2 Large Language Models

Large Language Models (LLMs) sind Sprachmodelle, welche in der Lage sind, natürliche Sprache zu interpretieren wie auch zu generieren. Es handelt sich um eine Kombination aus **Natural Language Processing (NLP)**, Deep Learning und generativer KI Modelle, die auf Basis großer Mengen an Textdaten trainiert werden. Da LLMs auf NLP basieren wenden sie das Prinzip der Tokenisierung an um Texte als Sequenz von Tokens darzustellen, welche wiederum kleinere Satzbausteine, wie Wörter und Satzzeichen, representieren. [3]
Seit der von Vaswani et al. eingeführte Transformer-Architektur [4], sind LLMs in der Lage Self-Attention anzuwenden, um Zusammenhänge zwischen allen Tokens parallel zu erfassen und jedem Token Gewichte zuzuweisen, die seine Relevanz in Bezug auf andere Tokens bestimmt. Dies erlaubt es LLMs komplexe sprachliche Strukturen und Zusammenhänge zu verstehen.
Die Leistungsfähigkeit eines LLMs hängt maßgeblich von der Anzahl und Struktur seiner Parameter ab. Parameter sind die lernbaren Gewichtungen innerhalb des neuronalen Netzes, welche während des Trainings optimiert werden, um Muster und Zusammenhänge in den Daten zu erkennen. Ein Modell mit einer höheren Kapazität verfügt über mehr Parameter und kann dadurch komplexere sprachliche Strukturen und semantische Abhängigkeiten erfassen. Allerdings geht mit wachsender Modellgröße auch ein steigender Speicher- und Rechenaufwand einher. [3]
Das Trainieren von LLMs lässt sich heutzutage typischerweise in drei Phasen aufteilen:

- **Pre-Training** beschreibt den Prozess das Modell auf umfangreichen und unspezifischen Texten zu trainieren, um grundlegende Sprachstrukturen, Grammatik wie auch Semantik zu erlenen. Das Ziel ist ein allgemeines Sprachverständnis zu entwickeln, welches eine Basis für vielfältige Anwendungen bietet.
- **Fine-Tuning** passt das vortrainierte Modell anschließend auf spezifische Aufgaben an. Dabei wird das Modell mit einem kleineren, aufgabenbezogenen Datensatz weiter trainiert. Dieser Prozess verfeinert und spezialisiert das generelle Wissen des Modells und ermöglicht so bessere Leistung für konkrete Anwendungsfälle.
- **Reinforcement Learning** ist ein zusätzliche Methode, welche dazu dienen soll ein Modell durch Interaktion mit einer Umgebung und basierend auf Belohnungssignalen sein Verhalten zu optimieren. Eine prominente Variante ist dabei **Reinforcement Learning from Human Feedback**, welche explizit menschliche Rückmeldungen nutzt, um hochwertigere und kontextpassendere Antworten von dem LLM zu erzielen.

Ein wesentliches Merkmal moderner LLMs ist ihre Fähigkeit zur Kontextverwaltung. Modelle verarbeiten Texte nicht isoliert, sondern berücksichtigen den bisherigen Verlauf einer Konversation. Dieser Kontext wird innerhalb eines Kontextfensters gespeichert, welches eine bestimte Anzahl von Tokens umfasst. Demnach ist das Gedächtnis eines LLMs stark von der Menge seiner verfügbaren Tokens abhängig. Die Genauigkeit eines LLMs beschreibt wie präzise es auf Eingaben reagiert und korrekte Informationen liefert. Sie ist stark von der Qualität der Trainingsdaten, Modellarchitektur und Optimierungsmethoden abhängig. LLMs neigen gelegentlich trotz jeder Leistungsfähigkeit dazu zu halluzinieren, wobei das Modell plausible, aber faktisch falsche Aussagen generiert, also Aussagen, die sprachlich schlüssig und kontextuell glaubwürdig erscheinen, inhaltlich jedoch nicht mit verifizierbaren Fakten übereinstimmen. Ein kontinuirliche Verbesserung der Genauigkeit ist zentral, um LLMs verlässlicher und damit insgesamt einsatzfähiger machen zu können. [3]
Ein Beispiel für den praktischen Einsatz, ist die E-Commerce-Branche. Durch die Fähigkeit natürliche Sprache zu generieren, wie auch zu analysieren, sind LLMs in der Lage mit Kunden produktiv zu interagieren. Dies bietet die Möglichkeit Kundenservices zu erweitern oder auch auf effizienter zu gestalten. [5] Durch zusätzliches Fine-Tuning lässt sich das Modell auf bestimmte Fachbereiche spezialisieren, um hochwertigere in einem eingeschränkterem Einsatzbereich zu erreichen.

2.3 Herausforderungen in der Praxis

Prozessmodellierung stellt in der Theorie, wie in dem Kapitel über die Grundlagen der Prozessmodellierung dargestellt, ein wirkungsvolles Mittel zur Analyse und Optimierung betrieblicher Abläufe dar. In der praktischen Anwendung zeigen sich jedoch verschiedene Herausforderungen.

Die konkreten Herausforderungen zeigen sich insbesonders in der praktischen Arbeit der Abteilung Produktionsmanagement des Werkzeugmaschinenlabors (WZL) der RWTH Aachen, deren zentrale Aufgabenfelder unter anderem in der methodischen Erfassung, Modellierung und Analyse industrieller Geschäftsprozesse liegen. Das hier zu entwickelnde Konzept und den dazugehörigen Prototypen adressieren daher direkte Anforderungen aus diesem operativen Umfeld. [6, 7]

Ein zentraler Bedarf ergibt sich aus der nachhaltigen Arbeit mit bestehenden Modellen und der Einarbeitung neuer Personen in komplexe Prozesslandschaften. Die Abteilung nutzt die eigenentwickelte Prozessmodellierungssprache Aixperanto und das Tool Process Flow, um Produktionsabläufe für Partnerunternehmen zu dokumentieren und zu analysieren. Diese Modelle müssen bei Prozessänderungen oder Optimierungen kontinuierlich gepflegt werden.

Für neue Mitarbeitende oder Kundenspezialisten stellt dies eine konkrete Herausforderung dar. Ein bestehendes, umfangreiches Process Flow-Modell, das in der Praxis weitaus komplexer sein kann als das zuvor vorgestellte Kundenauftragsbeispiel, ist schwer zugänglich. Die herkömmliche Interaktion über grafische Diagramme erlaubt keine gezielten Nachfragen zu bestimmten Aktivitäten, Ereignissen oder Bedingungen. Dieser fehlende intuitive Zugang verlängert die Einarbeitungszeit und erschwert die gemeinsame Arbeit an den Modellen. Zudem behindert die bestehende Schnittstelle die langfristige Pflege. Anpassungen aufgrund von Prozessverbesserungen müssen manuell mit spezifischem Modellierungswissen in Process Flow vorgenommen werden. Diese hohe Anforderung an den Nutzer kann dazu führen, dass notwendige Aktualisierungen verzögert werden und die Modelle an Aktualität verlieren.

Bereits im Vorfeld wurde innerhalb der Abteilung Arbeit durchgeführt, die darauf abzielte, die Fähigkeiten von LLMs für die Prozessmodellierung nutzbar zu machen. Ein Ansatz fokussierte sich auf die Unterstützung des initialen Modellierungsschritts. Hierbei wurde ein anonymisiertes, reales Prozessmodell verwendet, das in seinem Umfang und seiner Komplexität das hier demonstrierte Kundenauftragsbeispiel um mehr als das Vierfache übertraf. Diese Arbeit zeigte das Potenzial von LLMs zur Strukturierung und Vorverarbeitung von Prozessbeschreibungen auf, adressierte jedoch lediglich die initiale Prozessaufnahme. [8]

In Anbetracht dieser Herausforderungen verfolgt das vorgestellte Ziel, die kontinuierliche Arbeit mit Prozessmodellen grundlegend zu unterstützen. Das KI-gestützte System soll es ermöglichen, dass sich Nutzer unabhängig von ihren Vorkenntnissen in bestehende Modellstrukturen einarbeiten und diese im Arbeitsalltag gezielt abfragen und aktuell halten können. Diese Erweiterung strebt an, die langfristige Nutzung von Process Flow zu optimieren und das Tool für vielfältigere Anwendungsfälle und Nutzergruppen innerhalb und außerhalb der Abteilung zugänglich zu machen.

3 Anforderungen

Aufbauend auf der Analyse der bestehenden Herausforderungen werden in diesem Kapitel die funktionellen und nicht-funktionellen Anforderungen systematisch hergeleitet. Diese Anforderungen dienen als Bewertungsmaßstab für die Ausgestaltung und Umsetzung des vorgestellten Ziels.

3.1 Funktionelle Anforderungen

In diesem Abschnitt werden die funktionellen Anforderungen beschrieben, die das System erfüllen muss, um die vorgesehenen Aufgaben und Anwendungsfälle zu unterstützen.

3.1.1 Zugriff auf Prozessmodelle

Das System muss in der Lage sein, auf Prozessmodelle und auf ihre einzelnen Elemente zugreifen zu können. Diese Anforderung bildet die technische Grundlage für alle nachfolgenden Funktionen und stellt sicher, dass der Chatbot auf konsistenten und vollständigen Daten arbeitet.

3.1.2 Anfragen in natürlicher Sprache

Der Chatbot muss Anfragen in natürlicher Sprache verstehen und sie als spezifische Zugriffe auf das Prozessmodell anwenden können. Diese Funktion stellt den Hauptnutzen der Lösung dar, da sie einen intuitiven Zugang zu Prozesswissen erlaubt.

3.1.3 Erläuterung von Prozessschritten

Das System muss in der Lage sein, einzelne Elemente des Prozessmodells verständlich zu beschreiben. Hierzu gehört die verständliche Darstellung von Aktivitäten, Rollenverantwortlichkeiten und Entscheidungspunkten. Dadurch können Nutzer nicht nur gezielte Anfragen stellen, sondern auch sich Kontext über das Prozessmodell in natürlicher Sprache erschließen können.

3.1.4 Bearbeitung von Prozessmodellen

Das System soll die Bearbeitung von Prozessmodellen direkt über natürlichsprachliche Eingaben ermöglichen. Nutzerinnen und Nutzer sollen Aktivitäten, Rollen, Ereignisse oder Verbindungen

hinzufügen, ändern oder entfernen können, ohne eine Modellierungsnotation beherrschen zu müssen. Änderungen müssen eindeutig interpretierbar sein, korrekt in die bestehende Modellstruktur integriert werden und unmittelbar im zugrunde liegenden Modell gespeichert werden.

3.1.5 Formale Modellanalyse

Das System soll grundlegende Analysefähigkeiten unterstützen, etwa das Erkennen von Inkonsistenzen oder unklaren Strukturen. Dadurch wird die Identifikation von Optimierungspotenzialen erleichtert und die Qualität der Prozessmodelle langfristig verbessert.

3.2 Nicht-Funktionelle Anforderungen

Neben den spezifischen Funktionen müssen qualitative Eigenschaften definiert werden, die über die gesamte Systemnutzung hinweg gewährleistet sein müssen. Diese nicht-funktionellen Anforderungen stellen sicher, dass das System praktisch einsetzbar und in die bestehende Umgebung integrierbar ist.

3.2.1 Verständlichkeit

Die Verständlichkeit der Systemantworten ist eine zentrale Anforderung für die praktische Nutzbarkeit. Antworten müssen nicht nur inhaltlich korrekt, sondern auch in ihrer sprachlichen und strukturellen Darstellung so gestaltet sein, dass sie von Nutzern ohne tiefgehende Expertise erfasst werden können.

3.2.2 Konsistenz

Das System muss Antworten liefern, die sich streng an den im Modell enthaltenen Informationen orientieren. Die Ausgaben dürfen weder spekulativ sein noch über den Modellinhalt hinausgehen. Zudem sollen gleiche oder ähnliche Anfragen in vergleichbaren Kontexten zu konsistenten Resultaten führen. Diese Kombination aus Korrektheit und Reproduzierbarkeit ist essenziell, um Vertrauen in das System aufzubauen und eine sachgerechte Nutzung sicherzustellen.

3.2.3 Skalierbarkeit

Das System soll in einer produktiven Umgebung skalierbar sein. Konkret muss die Architektur ein Wachstum der Nutzerbasis sowie der Anfragelast unterstützen ohne dass die Verfügbarkeit beeinträchtigt wird.

3.2.4 Integrierbarkeit in das bestehende System

Das System muss sich nahtlos in die bestehende Architektur integrieren lassen. Dies umfasst sowohl die technische Anbindung an die vorhandenen Schnittstellen als auch die konsistente Einbettung in die Benutzeroberfläche. Die Integration soll ohne wesentliche Änderungen an der Kernlogik von Process Flow erfolgen und bestehende Daten sowie Prozessmodelle vollständig kompatibel halten. Das Ziel ist eine Erweiterung, die als natürlicher Bestandteil des bestehenden Ökosystems wahrgenommen wird.

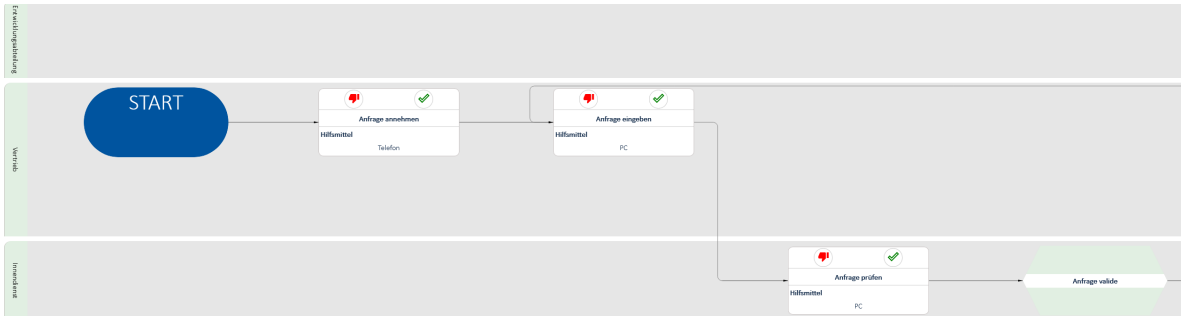
4 Verwendete Technologien

Dieses Kapitel stellt die für die Umsetzung des Ziels ausgewählten Technologien dar und erläutert die Gründe für ihre Auswahl. Dabei wird insbesondere auf die Integration der Komponenten, ihre Skalierbarkeit, Effizienz und die praktische Anwendbarkeit im Kontext der Prozessmodellierung eingegangen.

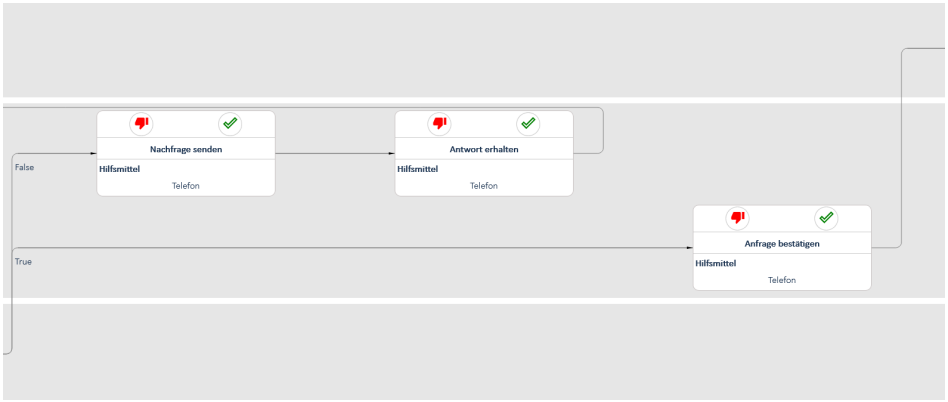
4.1 Aixperanto & Process Flow

Die Modellierungssprache Aixperanto bildet die konzeptionelle Grundlage für die in Process Flow erstellten Prozessmodelle. Aixperanto ermöglicht die Darstellung sowohl linearer Abläufe als auch verzweigter Prozessstrukturen einschließlich der in Unterkapitel 2.1 beschriebenen Grundelemente wie Aktivitäten, Ereignisse, Gateways und Rollen.
Process Flow ist eine webbasierte Anwendung zur strukturierten Modellierung, Dokumentation und Analyse von Produktionsprozessen. Die fachliche Entscheidung, diese Technologie zu verwenden, liegt in ihrem Ziel, Prozesswissen nicht nur zu dokumentieren, sondern als aktive Grundlage für Analyse und für die kontinuierliche Verbesserung nutzbar zu machen. Die Kombination aus tabellarischer Dateneingabe und grafischer Prozessvisualisierung sowie Bearbeitung ermöglicht eine präzise, konsistente und zugleich praxisnahe Abbildung realer Abläufe.
Die institutionelle Entscheidung für Process Flow und Aixperanto ergibt sich aus dem Arbeitskontext der Abteilung Produktionsmanagement des WZL der RWTH Aachen. Beide Technologien wurden innerhalb der Abteilung entwickelt und sind als Standardwerkzeuge fest in deren Arbeitsprozesse und Methodik verankert. Die Verwendung dieser Technologien ermöglicht also eine, wie in 3.2.4 beschriebende, nahtlose Integration in die bestehende Infrastruktur.

Die grafische Darstellung der Prozesse erfolgt in Form von Swimlane-Diagrammen (siehe Abbildung 4.1), die den Ablauf eines Prozesses entlang Akteuren oder organisatorischer Einheiten visualisieren. Jede Lane repräsentiert eine Rolle oder Abteilung, wodurch Verantwortlichkeiten und Schnittstellen unmittelbar sichtbar werden. Die einzelnen Prozessschritte, welche sowohl Aktivitäten, Ereignisse wie auch Start- oder Endknoten repräsentieren, werden als kartenbasierte Elemente dargestellt, die über gerichtete Pfeile (Sequenzflüsse) miteinander verknüpft sind und so den logischen Ablauf des Prozesses abbilden.



(a) Erster Teil des Beispielprozesses als Swimlane-Diagramm



(b) Zweiter Teil des Beispielprozesses als Swimlane-Diagramm

(c) Dritter Teil des Beispielprozesses als Swimlane-Diagramm

Abbildung 4.1: Swimlane-Diagramm des Beispielprozesses in Process Flow

Process Flow basiert auf einer modernen, mehrschichtigen Webarchitektur, bestehend aus einem Vue.js-Frontend, einem Java-Spring-Backend und einer relationalen Datenbank. Als Teil von Miori-Tools, welche eine Suite verschiedener Tools der Abteilung ist, nutzt Process Flow einen zentralen Keycloak-Server für die Authentifizierung und Autorisierung. Das Frontend generiert auf Basis der vom Backend bereitgestellten Daten nicht nur die bereits erwähnten grafischen Swimlane-Diagramme, sondern auch tabellarische Eingabeformen (siehe Abbildung 4.2) und ermöglicht eine reaktive und benutzerfreundliche Interaktion. Das Backend übernimmt zentrale Aufgaben wie Validierung, Datenpersistenz, Projektverwaltung und die Bereitstellung der Prozesslogik. Dadurch entsteht ein klar strukturiertes System, das sowohl wartungsfreundlich als auch erweiterbar ist.

Prozessschritt	Typ	Hintergrundfarbe	Vorgänger	Beschreibung	Vertrieb	Wertschöpfung	Standardisier
1	PROZESS		11	Anfrage annehmen	Vertrieb	Verschwendung	
2	PROZESS		1 10	Anfrage eingeben	Vertrieb	Verschwendung	
3	PROZESS		2	Anfrage prüfen	Innendienst	Verschwendung	
5	PROZESS		4	Anfrage bestätigen	Vertrieb	Verschwendung	
6	PROZESS		4	Nachfrage senden	Vertrieb	Verschwendung	
7	PROZESS		5	neue Anfrage erhalten	Entwicklungsabteilung	Verschwendung	
4	ENTSCHEIDUNG		3	Anfrage valide	Innendienst	Verschwendung	

Neu

Abbildung 4.2: Ausschnitt der Tabelle des Beispielprozesses in Process Flow

Diese technische Grundlage ist insbesondere im Kontext dieser Arbeit relevant, da die Integration von **MCP** sowie die eines Chatbots, eine saubere Entkopplung zwischen Benutzeroberfläche, Geschäftslogik und externen KI-Systemen erfordert. Process Flow bietet hierfür eine geeignete Architektur, die eine solche Erweiterung technisch konsistent ermöglicht, was fachlich ebenfalls für Process Flow spricht.

4.2 Large Language Model-Schnittstelle

Für die Integration eines Large Language Models (LLM) in die Process Flow-Umgebung bedarf es einer definierten Schnittstelle. Im Rahmen dieser Arbeit fiel die Wahl auf die Ollama-API als einheitliche Integrationsschicht.
Die technische Entscheidung für die Ollama-API ist vornehmlich durch das Architekturprinzip der Schnittstelle motiviert, da sie eine standardisierte, REST-basierte Schnittstelle bereitstellt, die mit einer Vielzahl von Open-Source-LLMs kompatibel ist. Diese Abstraktion entkoppelt die Anwendungslogik des MCP-Clients im Process Flow-Frontend von der konkreten Implementie-

rung des KI-Backends. In der Praxis bedeutet dies, dass ein Wechsel des zugrundeliegenden Sprachmodells, etwa von Llama zu einem GPT oder DeepSeek Modell, keine Anpassungen im Client-Code erfordert, sondern lediglich eine Konfigurationsänderung darstellt. [9]
Die Ollama-API unterstützt mehrere Betriebsmodi, die den unterschiedlichen Phasen des Projekts und den verfügbaren Ressourcen gerecht werden. Für die Prototypenentwicklung und Evaluation ist der **lokale Betrieb** von entscheidendem Vorteil. Er ermöglicht Tests mit voller Datenkontrolle ohne laufende Betriebskosten. Diese Kontrolle ist insbesondere im wissenschaftlichen Kontext und beim Umgang mit potenziell sensiblen Prozessdaten aus Partnerunternehmen von großem Wert. Parallel dazu ist die Architektur für einen **Remote-Betrieb** mittels API-Key verfügbar. Dieser Modus würde es ermöglichen, rechenintensivere Modelle auf dedizierter Hardware auszuführen oder den Dienst in eine produktive Cloud-Umgebung zu skalieren, ohne die grundlegende Systemarchitektur ändern zu müssen. Die API-Schnittstelle bleibt in beiden Fällen identisch, was einen nahtlosen Übergang zwischen den Betriebsarten erlaubt. [9]
Die Ollama-API wurde als LLM-Schnittstelle gewählt, weil sie als einheitliche Abstraktionsschicht eine flexible Modellauswahl bietet, mehrere praktische Betriebsmodi (lokal/remote) unterstützt und damit der Anforderung der Integrierbarkeit wie auch eine letztendliche Entwicklung eines Prototyps ermöglicht.

4.3 MCP

Das Model Context Protocol (MCP) ist ein offenes Protokoll, das eine standardisierte Schnittstelle zwischen LLMs und externen Datenquellen bzw. Werkzeugen definiert. [10] Es wurde von Anthropic initiiert, um den mangelnden Zugriff auf aktuelle oder private Informationen und das statische Weltwissen von LLMs zu adressieren. [11] Somit ermöglicht das Protokoll, LLMs dynamisch mit spezifischem Kontext aus vertrauenswürdigen Quellen zu versorgen, ohne dass das zugrundeliegende Modell neu trainiert oder modifiziert werden muss. Dieser Ansatz erlaubt es, generische Sprachmodelle sicher und kontrolliert in spezifische Anwendungsfälle einzubetten.
In dieser Arbeit dient MCP als zentrale Schnittstelle zwischen dem LLM und den Prozessmodellen. Das primäre Ziel dieser Implementierung ist es, den für die Analyse und Bearbeitung benötigten Prozesskontext strukturiert und konsistent aufzubereiten. Dadurch wird die Grundlage geschaffen, sodass das LLM gezielte Abfragen beantworten und kontextsensitive Bearbeitungsvorschläge für Prozessmodelle generieren kann.
MCP folgt einer **Client-Server-Architektur**, die über das JSON-**Remote-Procedure-Call**-Protokoll (RPC) kommuniziert. Diese Architektur gewährleistet Modularität und Sicherheit, da verschiedene Kontextquellen durch separate Server bereitgestellt werden können, ohne den MCP-Client oder das LLM selbst zu modifizieren. Der **Host** behandelt die direkte Interaktion

mit dem Nutzer und hostet das LLM wie auch einen oder mehrere MCP-Clients. Der **MCP-Client** arbeitet dabei nur auf Protokollebene und koordiniert die Nutzeranfragen des Hosts, entscheidet basierend auf dem Prompt, welche Kontextinformationen benötigt werden, und sendet entsprechende Anfragen an die konfigurierten MCP-Server. Der **MCP-Server** stellt den Zugriff auf konkrete Fähigkeiten und Daten bereit. Ein Server ist ein eigenständiger Prozess, der für eine spezifische Ressource zuständig ist, etwa für ein lokales Dateiverzeichnis, eine SQL-Datenbank oder eine externe API. Der Server reagiert ausschließlich auf die Anfragen des Clients, führt die angeforderte Operation, wie z.B. das Lesen einer Datei oder Ausführen einer Datenbankabfrage, durch und sendet das Ergebnis im JSON-RPC Format zurück. [10]
Das Protokoll lässt sich zudem in zwei Ebenen aufteilen. Die **Data Layer** implementiert die auf JSON-RPC 2.0 basierte Semantik und Struktur des Austausches. Zudem beinhaltet diese Ebene eine Reihe zentraler Features. Das **Lifecycle Management**, welches die Verbindungsinitialisierung, Aushandlung der Verbindungsfähigkeit und die Verbindungstrennung verwaltet. Die Abstraktionen, welche auf Serverseite implementiert und von Clients verwendet werden können:

- **Resources** für einen strukturierten Lesezugriff
- **Tools** für die Ausführung von Aktionen
- **Prompts** für vordefinierte Dialoge

Die Abstraktionen, welche auf Clientseite implementiert und von Servern verwendet werden können:

- **Sampling** erlaubt es, eine Vervollständigung über das LLM des Hosts anzufragen.
- **Elicitation** erlaubt es, zusätzliche Informationen von dem Nutzer zu fordern.
- **Logging** erlaubt Logs an den Client weiterzuleiten.

Die **Transport Layer** verwaltet die Kommunikation und Authentifizierung zwischen Clients und Servern. Die Ebene unterstützt Stdio, welches für direkte lokale Prozesse verwendet werden kann, wie auch Streamable-HTTP mit optionalen Server-Sent Events für Remote-Verbindungen. [10]

5 Konzeption & Prototypisierung

Dieses Kapitel beschreibt die konzeptionelle Entwicklung und prototypische Implementierung. Das vollständige Konzept der einzelnen systemkomponenten wird in einzelnen Unterkapiteln beschrieben. Zum Schluss jedes Unterkapitels wird die prototypische Implementierung der jeweiligen Komponente dargestellt.
Die prototypische Umsetzung folgte einem Entwicklungsansatz, bei dem zunächst ein Minimal Viable Product mit grundlegenden Funktionen implementiert wurde, um die Machbarkeit des Konzepts zu validieren, wobei besonderer Wert auf die praktische Integrierbarkeit in das bestehende System gelegt wurde.

5.1 Systemarchitektur

Dieser Abschnitt konkretisiert die Architekturentscheidungen für die Integration von MCP in Process Flow und definiert die resultierende Zielarchitektur des Gesamtsystems und den Ablauf einer Interkation mit dem definierten System.

5.1.1 Zielarchitektur und Komponentenintegration

Die vorangegangenen Abschnitte 4.3 und 4.1 haben einerseits die Zielarchitektur eines MCP-basierten Systems und andererseits die bestehende Architektur von Process Flow dargestellt. Der konzeptionelle Integrationsansatz besteht darin, die einzelnen MCP-Komponenten systematisch in die Process Flow-Architektur einzubetten, um eine konsistente Gesamtarchitektur zu erreichen, die den MCP-Spezifikationen entspricht.
Ausgehend von der MCP-Systemarchitektur lässt sich das Process Flow-Frontend als Host-Komponente identifizieren, da diese für die direkte Nutzerinteraktion verantwortlich ist. Diese Zuordnung ergibt sich aus der zentralen Rolle des Frontends als primäre Benutzerschnittstelle, die alle Nutzereingaben entgegennimmt und Systemantworten präsentiert. Dementsprechend wird der MCP-Client innerhalb des Frontends implementiert, da dieser als Vermittler zwischen Host und MCP-Server fungiert. Parallel dazu wird das LLM an den Client angebunden, um die natürlichsprachliche Verarbeitung der Nutzereingaben zu ermöglichen. Damit sind die Positionen der drei zentralen MCP-Komponenten Host, Client und LLM innerhalb der Process Flow-Architektur eindeutig bestimmt.
Die konzeptionelle Herausforderung besteht nun in der Anbindung des MCP-Systems an eine

geeignete Service- oder Datenquelle. Grundsätzlich benötigt das LLM bidirektionalen Zugriff auf Prozessmodelle, um sowohl lesende als auch schreibende Operationen durchführen zu können. Es ergeben sich zwei Varianten: eine Anbindung an die Datenbank oder an das Process Flow-Backend.
Prinzipiell lassen sich beide Varianten technisch durch einen MCP-Server realisieren. Es bietet sich jedoch an, einen dedizierten MCP-Server für das Process Flow-Backend zu implementieren, da es bereits etablierte Sicherheitsmechanismen und Datenvalidierungen bietet, die bei direkten Datenbankzugriffen nicht gegeben wären.
Die resultierende Systemarchitektur zeigt Abbildung 5.1. Das Frontend beherbergt sowohl den MCP-Client als auch die LLM-Integration und kommuniziert über das MCP-Protokoll mit dem dedizierten MCP-Server. Dieser übersetzt die MCP-Requests in Backend-API-Aufrufe und stellt sicher, dass alle Operationen im Kontext des authentifizierten Benutzers ausgeführt werden. Das Backend bleibt als zentrale Geschäftslogik erhalten und gewährleistet Datenkonsistenz und Sicherheit.

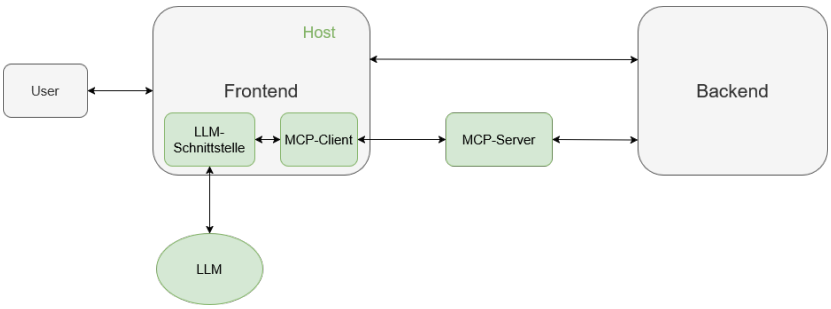


Abbildung 5.1: Abbildung der Zielarchitektur

5.1.2 Interaktionsablauf

Die Systeminteraktion folgt einem sequentiellen Prozess, der sich in sechs klar definierte Verarbeitungsschritte beschreiben lässt. Abbildung 5.2 visualisiert diesen Ablauf als Sequenzdiagramm, das die Kommunikation zwischen den Systemkomponenten und deren zeitliche Abfolge darstellt.

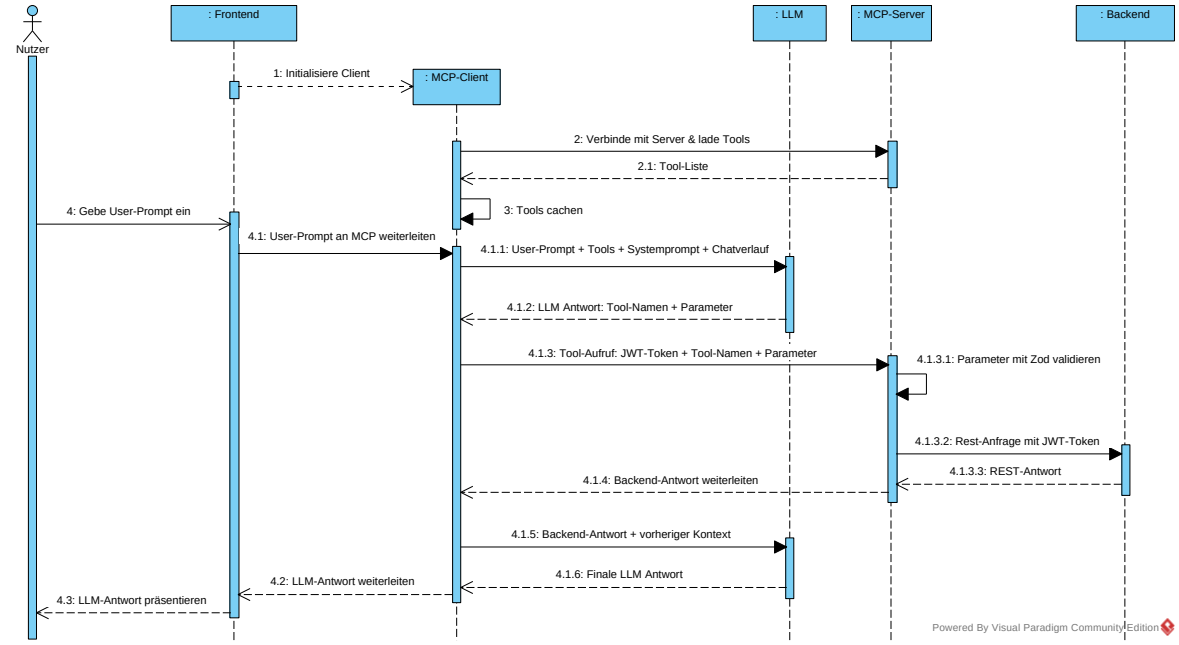


Abbildung 5.2: Abbildung des Interaktionsablaufs als Sequenzdiagramm

1. **Initialisierungsphase**: Das Frontend initialisiert den MCP-Client, der eine Verbindung zum MCP-Server aufbaut. Der Server antwortet mit einer Liste aller verfügbaren Tools und deren Parameter-Schemata, die der Client lokal zwischenspeichert. Zuvor authentifiziert sich der Benutzer über Keycloak und erhält einen JWT-Token, der an den MCP-Client übergeben wird.

2. **Eingabeverarbeitung**: Der Benutzer formuliert eine natürlichsprachliche Anfrage im Chat-Interface. Das Frontend leitet diese Eingabe an den MCP-Client weiter, der sie mit den zwischengespeicherten Tool-Beschreibungen anreichert und als strukturierten Prompt an das Llama 3.1-Modell sendet.

3. **Semantische Analyse und Tool-Selektion**: Das LLM analysiert den Prompt, iden-

tifiziert die Benutzerintention und extrahiert relevante Parameter. Es generiert eine JSON-Struktur mit Tool-Name und Parametern, die an den MCP-Client zurückgesendet wird. Der Client validiert diese Struktur gegen die gespeicherten Tool-Schemata.

4. **Tool-Execution**: Nach erfolgreicher Validierung sendet der MCP-Client einen Tool-Execution-Request an den MCP-Server. Dieser Request enthält die validierten Parameter und den JWT-Token im Authorization-Header. Der MCP-Server extrahiert den Token für die weitere Verarbeitung.

5. **Call an das Backend**: Der MCP-Server transformiert den MCP-Request in einen entsprechenden REST-API-Aufruf an das Spring-Boot-Backend, wobei der JWT-Token mitgeführt wird. Das Backend validiert den Token via Keycloak, führt die angeforderte Operation in der Datenbank aus und sendet das Ergebnis zurück an den MCP-Server.

6. **Antwortgenerierung**: Der MCP-Server formatiert die Backend-Antwort gemäß MCP-Spezifikation und sendet sie an den MCP-Client. Dieser setzt die Antwort in den Kontext des LLMs ,welcher eine finale Antwort generiert, die im Chat-Interface präsentiert wird.

5.2 Konzeption & Prototypisierung der Systemkomponenten

Es wird ein Prototyp entwickelt, der als Proof of Concept dient und gleichzeitig die Grundlage für die Validierung des Konzepts bildet. Ein funktionaler Minimalprototyp, der die Kerninteraktion demonstriert, genügt für die Validierung des grundlegenden Ansatzes. Grundlegend entspricht der Prototyp der Konzeption. Die wesentliche Differenz zwischen Prototyp und vollständigem Konzept liegt im Umfang und der Tiefe der Implementierung. Während das Konzept alle gewünschten Anforderungen tiefgreifend abbilden soll, konzentriert sich der Prototyp auf die essentiellen Funktionen, die für einen funktionierenden Workflow notwendig sind und implementiert damit eine reduzierte Version der Konzeption.

5.2.1 Host

Das Process Flow-Frontend übernimmt die Rolle des Hosts im MCP-Protokoll und muss entsprechend erweitert werden, um die MCP-Spezifikation vollständig zu erfüllen. Darunter zählt ein Nutzerinterface zur Erfassung und Präsentation von Nutzerprompts und Systemantworten, ein LLM zur KI-basierten Verarbeitung natürlicher Sprache, sowie ein MCP-Client als zentrale Vermittlungskomponente zwischen den anderen Systemteilen.

5.2.1.1 Nutzerinterface

Konzeptionierung

Das Nutzerinterface, eine Vue.js-Komponente für eine ideale Einbindung in das bestehende Frontend, ist als Reiter innerhalb der Projektansicht erreichbar und bietet eine Chatbot-Oberfläche für die Interaktion mit dem MCP-System. Das Ziel ist es ein Design zu bieten, welches sich an etablierte Chat-Interfaces orientiert, um eine klare, strukturierte Darstellung von Nutzeranfragen und Systemantworten in chronologischer Abfolge zu gewährleisten.
Im Idealfall umfasst die Funktionalität neben der grundlegenden Eingabe und Ausgabe von Nachrichten erweiterte Interaktionsmöglichkeiten zur Verbesserung der Nutzererfahrung. Dazu gehört die Möglichkeit, laufende Anfragen durch den Nutzer abzubrechen, die visuelle Kennzeichnung von in Bearbeitung befindlichen Antworten, das Erstellen neuer Sessions und das einfache Wechseln zwischen verschiedenen Sessions. Diese Feature unterstützen eine effiziente und intuitive Nutzung des Systems auch in komplexen Arbeitskontexten.

Prototypisierung

Die Implementierung im Prototypen beschränkt sich auf eine einfache Chatbot-Oberfläche, die die grundlegende Interaktion ermöglicht. Auf erweiterte Features wie die Verwaltung mehrerer Sessions oder den Abbruch laufender Anfragen wurde verzichtet, um den Implementierungsaufwand zu reduzieren und den Fokus auf die Kernfunktionalität zu legen.

5.2.1.2 MCP-Client

Konzeptionierung

Der MCP-Client wird als TypeScript-Service innerhalb des Process Flow-Frontends implementiert und basiert dabei auf dem offiziellen TypeScript SDK für MCP-Clients. Seine Implementierung übernimmt die zentralen Aufgaben zur Steuerung der KI-Interaktion. Dies umfasst den Aufbau und die Verwaltung der Verbindung zum MCP-Server. Bei der Initialisierung empfängt der Client eine vollständige Liste aller verfügbaren Tools inklusive ihrer Parameter-Schemata vom Server und speichert diese Informationen lokal zwischen. Für die Ausführung von Tool-Calls validiert er die vom LLM generierten Aufrufe gegen diese zwischengespeicherten Schemata und leitet sie an den Server weiter. Zudem stellt der Client eine definierte Schnittstelle für das Vue.js-Nutzerinterface bereit, über die das Chat-Modul neue Nutzeranfragen initiieren und Systemantworten asynchron empfangen kann.
Eine wesentliche konzeptionelle Anforderung ist die Weiterleitung des Benutzer-JWT-Tokens an den MCP-Server, um authentifizierte Backend-Requests zu ermöglichen. Zusätzlich implementiert der Client die zwei Abstraktionen Sampling und Elicitation, um Antworten und damit das allgemeine Nutzererlebnis, umso mehr bei komplexeren Anfragen, zu verbessern.

Prototypisierung

Bei der Implementierung des Prototyps wurde auf eine Umsetzung dieser Abstraktionen verzichtet. Der Fokus lag stattdessen auf der Implementierung der grundlegenden Funktionalitäten, die für einen vollständigen Interaktionsablauf zwischen Nutzer, LLM, MCP-Server und Backend erforderlich sind. Dieser reduzierte Ansatz ermöglichte eine effiziente Validierung des Kernkonzepts, bevor in einer möglichen vollständigen Implementierung erweiterte Features wie Sampling und Elicitation integriert werden.

5.2.1.3 LLM

Konzeptionierung

Die Integration erfolgt über die in Kapitel 4.2 beschriebene einheitliche Schnittstelle, welche durch die Ollama-API gegeben ist und somit Kompatibilität mit verschiedenen LLM-Backends gewährleistet. Diese Abstraktion ermöglicht den Austausch des zugrundeliegenden Modells ohne Anpassungen am Client-Code und unterstützt die Evolution der KI-Komponente im Laufe der Systementwicklung. Die Schnittstelle definiert klar spezifizierte Methoden für Prompt-Verarbeitung, Response-Generierung und Kontextmanagement, die der MCP-Client zur semantischen Analyse von Nutzereingaben nutzt.

Prototypisierung

Der entwickelte Prototyp nutzt das Large Language Model Llama 3.1 mit 8B Parametern, das lokal gehostet wird. Die Wahl dieses Modells fiel aus mehreren praktischen Gründen. Es bietet eine umfassende Kompatibilität mit der verwendeten Ollama-API. Darüber hinaus hat sich dieses Modell in vorherigen Forschungsarbeiten zum Einsatz von LLMs in der Prozessmodellierung als geeignete Basis erwiesen [8].

5.2.2 MCP-Server

Konzeptionierung

Der MCP-Server wird ebenfalls wie der MCP-Client mithilfe der offiziellen MCP TypeScript SDK implementiert. Seine konzeptionelle Hauptaufgabe besteht in der Bereitstellung einer sicheren Schnittstelle zwischen dem MCP-Client und dem Spring Boot Backend. Der Server übersetzt MCP-spezifische Requests in entsprechende Backend-API-Aufrufe und gewährleistet dabei die Integrität und Sicherheit der Datenübertragung.
Die Tool-Implementierung zur Abdeckung konkreter Geschäftslogiken über das Backend, folgt einem strukturierten Schema-basierten Ansatz, bei dem jedes Tool seine Parameter mittels Zod-Schemata definiert und validiert. Diese Validierung erfolgt sowohl durch Datentypen, Formate als auch durch Wertebereiche, Abhängigkeiten. Zudem implementiert der Server

prompts zur flexiblen Handhabung komplexerer oder weniger strukturierter Anfragen. Diese Dualität ermöglicht eine ausgewogene Balance zwischen präziser, typsicherer Funktionalität und erweiterbarer Interaktion.
Die Authentifizierung erfolgt als transparente Token-Weiterleitung, wobei der Server JWT-Bearer-Tokens vom Client empfängt, validiert und unverändert an das Backend weiterleitet. Dieser Ansatz implementiert das Zero-Trust-Prinzip, bei dem der Server selbst keine Token speichert oder verarbeitet, sondern lediglich als sicherer Proxy fungiert, während die eigentliche Autorisierungsentscheidung beim Backend verbleibt.
Das Fehlerbehandlungskonzept des Servers sieht eine mehrstufige Architektur vor, die verschiedene Fehlertypen differenziert behandelt. Validierungsfehler auf Protokollebene führen zu strukturierten MCP-Fehlermeldungen mit konkreten Hinweisen zur Problembehebung. Backend-Fehlermeldungen werden in das MCP-Protokollformat übersetzt und mit sinnvollem Kontext angereichert, um dem Nutzer aussagekräftige Informationen zu liefern. Interne Serverfehler werden protokolliert und durch generische Fehlermeldungen abgefangen, die keine sensiblen Systeminformationen preisgeben.

Prototypisierung

In der prototypischen Implementierung wurde der MCP-Server auf eine begrenzte Anzahl von Tools und Prompts reduziert. Die Tool-Palette beschränkt sich auf essenzielle Operationen zur Demonstration des Konzepts, darunter das Auflisten von Projekten, das Erstellen einfacher Prozesse und das Hinzufügen von Prozessschritten. Eine vollständige Abdeckung der Backend-Geschäftslogik wurde bewusst nicht angestrebt. Der Server leitet Authentifizierungs-Tokens unverändert an das Backend weiter, implementiert aber keine erweiterte Token-Validierung oder Refresh-Logik. Die Fehlerbehandlung implementiert grundlegende Mechanismen zur Abfangen von Validierungsfehlern und Netzwerkproblemen, verzichtet jedoch auf umfangreiche detaillierte Benutzerrückmeldungen.

6 Evaluation

Dieses Kapitel stellt die Überprüfung der entwickelten Konzeption und des Prototyps gegen die in Kapitel 3 definierten funktionalen und nicht-funktionalen Anforderungen dar. Der implementierte Prototyp repräsentiert eine reduzierte, aber vollständige Umsetzung der Kernkonzepte und dient somit als zentrale Testinstanz für die praktische Validierung und ermöglicht so eine aussagekräftige Bewertung der Machbarkeit und der grundlegenden Systemeigenschaften.
Die Evaluation folgt einer strukturierten Methodik. Zunächst werden im ersten Unterkapitel konkrete Testszenarien durchgeführt, die die wesentlichen Use-Cases die in 3.1 beschriebenen funktionalen Anforderungen abdecken. Die generierten Systemantworten und das Verhalten des Prototyps während dieser Tests bilden die empirische Grundlage für die anschließende Bewertung in dem zweiten Unterkapitel. Dort werden die Ergebnisse analysiert, um sowohl die Erfüllung der funktionalen Anforderungen als auch die Ausprägung der in Kapitel 3.2 nicht-funktionalen Anforderungen zu beurteilen.
Als einheitliche Testbasis für alle Evaluationsschritte dient das bereits vorgestellte, repräsentative Prozessmodell des Kundenauftragszyklus. Dieses Projekt bildet einen repräsentativen Anwendungsfall ab und gewährleistet damit eine konsistente Bewertung aller Systemfunktionen.

6.1 Tests & Ergebnisse

Zugriff auf Prozessmodelle wie auch **Anfrage in natürlicher Sprache** stellen beide die fundamentale Basis für alle weiteren Systemfunktionen dar. Die Evaluation dieser Anforderungen konzentriert sich darauf, zu überprüfen, ob das System einerseits auf die Prozessmodelle zugreifen kann und ob dies auch der Nutzer in der Form von natürlicher Sprache anfragen kann. Die Testdurchführung lässt sich repräsentativ für beide Anforderungen in einem Test zusammenfassen. Durch die Anfrage aller Prozessschritte des Beispielprozesses, wie in Abbildung 6.1 gezeigt, lässt sich einerseits feststellen, ob das System auf das Prozessmodell zugriff hat und ob es andererseits die Anfrage des Nutzers in natürlicher Sprache richtig interpretieren und anwenden kann.

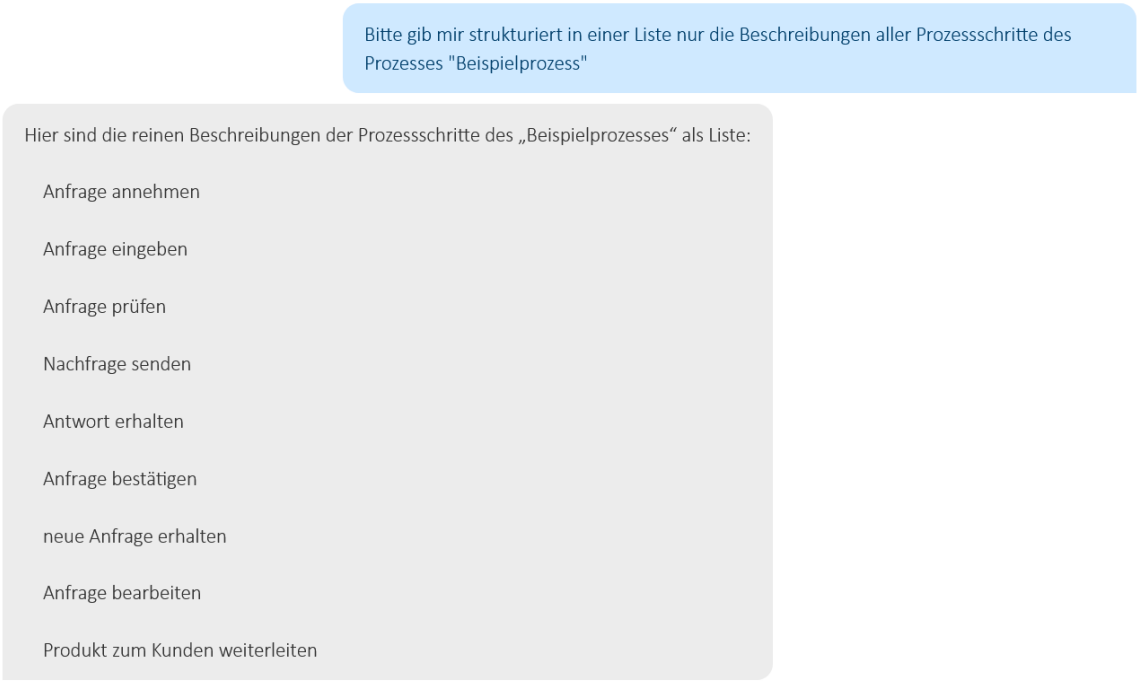


Abbildung 6.1: Anfrage zum testen des Zugriffsfähigkeit und Anfragenverarbeitung in natürlicher Sprache

Die **Erläuterung von Prozessschritten** zielt darauf ab, die Verständlichkeit und Vollständigkeit der generierten Beschreibungen von Prozesselementen zu prüfen. Getestet wurden Beschreibungen von Aktivitäten, Rollenverantwortlichkeiten und Entscheidungspunkten, wie in der folgenden Abbildung 6.2 im Fall der Rollenverantwortlichkeiten und in Abbildung 6.3 im Falle einer Aktivität und einer Entscheidung abgebildet.

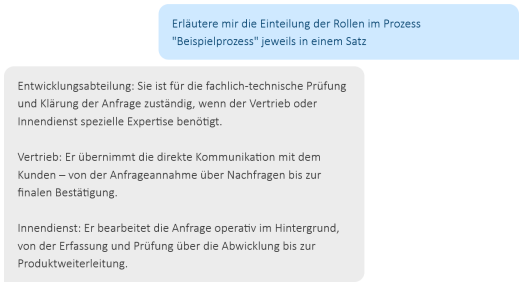


Abbildung 6.2: Anfrage zum testen der Erläuterung von Rollen

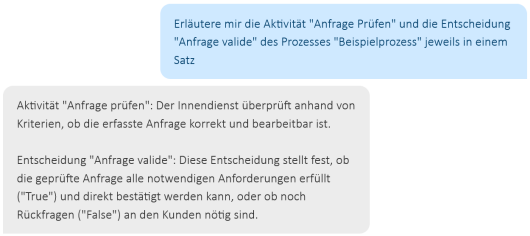


Abbildung 6.3: Anfrage zum testen der Erläuterung von einer Aktivität bzw. Entscheidung

Bearbeitung von Prozessmodellen evaluiert die Fähigkeit des Systems, Prozessmodelle über natürliche Spracheingaben zu modifizieren. Die Tests konzentrieren sich auf die drei Grundoperationen Hinzufügen, Ändern und Entfernen von Prozesselementen. Der Test umfasst stichprobenartig spezifische Bearbeitungsszenarien, die jeweils eine natürlichsprachliche Anweisung und das erwartete Ergebnismodell definieren. Dafür wird, wie in dieser Abbildung 6.4 gezeigt, ein Prozessschritt "Testïn dem Beispielprozess hinzufügt, zu "Test Fooümbenannt und daraufhin gelöscht.

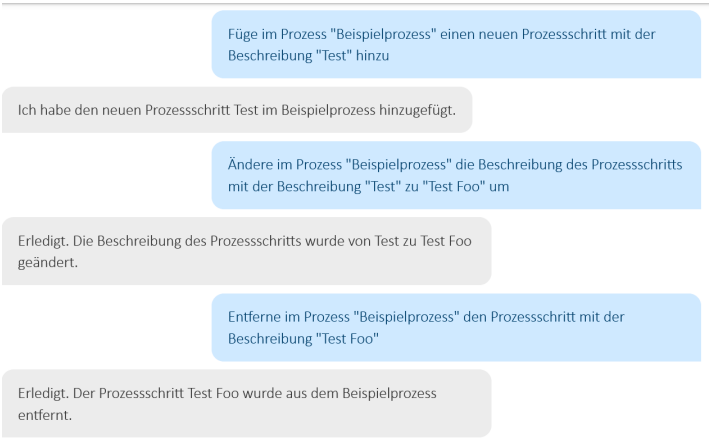


Abbildung 6.4: Anfragen zum testen der Bearbeitungsfähigkeit von Prozessmodellen

Formale Modellanalyse bewertet die grundlegenden Analysefähigkeiten des Systems zur Identifikation von Inkonsistenzen und Optimierungspotenzialen in Prozessmodellen. Die Evaluation konzentriert sich auf die Erkennung definierter Problemkategorien. Für den Test wurde demonstrativ das Hilfsmittel in einer der Prozessschritte offen gelassen, sodass das System dies wie in Abbildung 6.5 benennt.

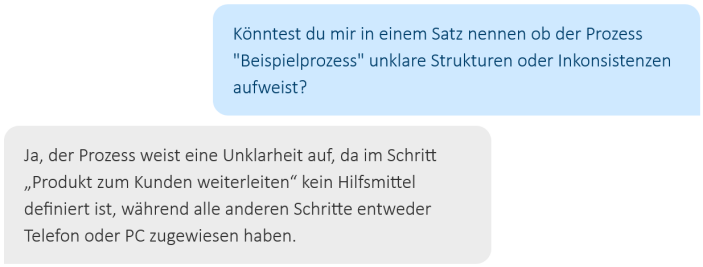


Abbildung 6.5: Anfragen zum testen der formalen Modellanalyse

6.2 Bewertung der Ergebnisse & des Konzepts

Die durchgeführten Tests liefern eine fundierte Grundlage für die Bewertung des prototypischen Systems gegen die spezifizierten Anforderungen. Die folgende Analyse fasst die Ergebnisse zusammen und bewertet den erreichten Reifegrad sowie identifizierte Grenzen des Konzepts.

6.2.1 Erfüllung der funktionalen Anforderungen

Die Evaluation zeigt, dass der Prototyp die grundlegenden funktionalen Anforderungen in ihrem Kern erfüllt.

- **Zugriff & Natürlichsprachliche Anfrage:** Die erfolgreiche Abfrage aller Prozessschritte demonstriert, dass die zentrale Architektur aus MCP-Client, -Server und Backend-Integration funktional ist. Das System kann auf Prozessmodelle zugreifen und natürlichsprachliche Eingaben in strukturierte Abfragen übersetzen.
- **Erläuterung von Prozessschritten:** Die generierten Beschreibungen für Aktivitäten, Rollen und Entscheidungspunkte belegen, dass das LLM in der Lage ist, strukturierte Daten vom Process Flow-Backend in verständlichen natürlichen Text zu transformieren. Die Qualität der Erläuterungen ist für eine erste Orientierung ausreichend, jedoch von der Qualität und Vollständigkeit der Modellmetadaten abhängig.
- **Bearbeitung von Prozessmodellen:** Die erfolgreiche Durchführung der Grundoperationen Hinzufügen, Ändern und Löschen beweist, dass der Bidirektionalitätsansatz des MCP-Protokolls praktisch umsetzbar ist. Der Nutzer kann Änderungen in natürlicher

Sprache anstoßen, die korrekt interpretiert und als gültige Transaktionen im Backend ausgeführt werden.

- **Formale Modellanalyse:** Die Erkennung des offen gelassenen Hilfsmittels zeigt, dass das System mit formalen Analysen umgehen und diese dem Nutzer kommunizieren kann. Dies stellt die Basis für erweiterte Analysefunktionen dar, auch wenn der Prototyp nur eine einzelne, vorher festgelegte Inkonsistenz prüft.

Insgesamt beweist der Prototyp die Machbarkeit des Kernkonzepts einer MCP-basierten, natürlichsprachlichen Schnittstelle für die Interaktion mit Process Flow-Modellen.

6.2.2 Bewertung nicht-funktionaler Eigenschaften

Anhand der Testantworten und des Systemverhaltens lassen sich Rückschlüsse auf die nicht-funktionalen Eigenschaften ziehen.

- **Verständlichkeit:** Die Antworten präsentieren komplexe Informationen wie Rollenzuordnungen oder Ablaufbedingungen in logisch gegliederten Sätzen, was die Orientierung im Antworttext erleichtert. Inwieweit die Antworten jedoch auch für Nutzer mit völlig unterschiedlichem Vorwissen über Prozessmodellierung im Allgemeinen gleichermaßen intuitiv verständlich sind, wurde im Rahmen der Evaluation nicht systematisch untersucht.
- **Konsistenz:** In den durchgeführten Tests blieben die Antworten stets innerhalb der Grenzen des verwendeten Prozessmodells und wiesen keine Halluzinationen auf. Dies ist auf die strikte Architektur zurückzuführen, bei der das LLM ausschließlich mit von den MCP-Tools bereitgestellten und validierten Daten arbeitet.
- **Skalierbarkeit:** Die Verwendung von MCP resultierte in einer Architektur, die eine horizontale Skalierung durch die unabhängige Skalierbarkeit ihrer entkoppelten Komponenten erlaubt. Dies ermöglicht es, das System an eine wachsende Nutzerbasis oder komplexere Anfragen anzupassen, ohne die Kernlogik zu verändern.
- **Integrierbarkeit:** Der Prototyp operierte erfolgreich innerhalb der bestehenden Process Flow-Infrastruktur, nutzte die Keycloak-Authentifizierung und griff auf die Backend-API zu. Dies belegt, dass der konzipierte Integrationsweg technisch umsetzbar ist, ohne die Kernlogik des bestehenden Systems zu modifizieren.

Die Skalierbarkeit und Performance unter Last konnten im Rahmen des Prototyps nicht evaluiert werden und bleiben ein Gegenstand für zukünftige Arbeiten.

6.2.3 Einschränkungen und zukünftiger Entwicklungsbedarf

Die Evaluation offenbart auch bewusste und unbewusste Grenzen des aktuellen Prototyps, die den Weg für zukünftige Arbeiten weisen.

- **Begrenzte Testtiefe:** Die Tests wurden an einem einzelnen, überschaubaren Beispielprozess durchgeführt. Die Leistungsfähigkeit bei sehr großen, komplexen oder unvollständigen Modellen sowie die Robustheit gegenüber mehrdeutigen oder fehlerhaften Nutzereingaben bleibt zu prüfen.
- **Reduzierte Funktionalität:** Wie konzeptionell vorgesehen, implementiert der Prototyp nur eine minimale Tool-Palette. Eine produktive Nutzung erfordert die Erweiterung um alle relevanten Operationen sowie um komplexere Analysefunktionen.
- **Fehlerbehandlung und Benutzerführung:** Der Prototyp bietet nur grundlegende Fehlermeldungen. Eine nutzerorientierte Dialogführung, die bei Unklarheiten nachfragt (Elicitation) oder Alternativen vorschlägt, wäre für die Praxis essenziell.

6.2.4 Fazit der Evaluation

Die Evaluation bestätigt, dass die grundlegende Prämisse der Arbeit tragfähig ist. Die Integration eines Large Language Models über das Model Context Protocol bietet einen praktikablen Weg, um eine natürlichsprachliche Schnittstelle für Process Flow zu schaffen. Der Prototyp demonstriert erfolgreich die Kerntätigkeiten des Abfragens, Erläuterns, Bearbeitens und Analysierens. Er erfüllt damit sein primäres Ziel als Proof of Concept. Die identifizierten Einschränkungen markieren keinen Systembruch, sondern den natürlichen Umfang eines Prototyps und definieren die Anforderungen für eine potenzielle produktive Implementierung.

7 Zusammenfassung & Ausblick

Die vorliegende Arbeit hatte zum Ziel, ein Konzept und einen Prototyp für einen Chatbot zu entwickeln, der eine natürlichsprachliche Interaktion mit Prozessmodellen in der bestehenden Process Flow-Umgebung der Abteilung Produktionsmanagement des WZL ermöglicht. Ausgangspunkt war die praktische Herausforderung, dass Prozesswissen in Process Flow für viele Nutzer schwer zugänglich ist und dessen Pflege einen hohen manuellen Aufwand erfordert.
Die konzipierte Lösung nutzt das Model Context Protocol (MCP) als zentrale, standardisierte Integrationsschicht. Diese Architektur koppelt ein Large Language Model (Llama 3.1) über eine einheitliche Ollama-Schnittstelle sicher mit dem Process Flow-Backend. Der entwickelte Prototyp setzt dieses Konzept in einer reduzierten, aber vollständigen Form um. Er demonstriert erfolgreich die Kerntätigkeiten wie auch die formale Analyse von Prozessschritten. Die Evaluation zeigt, dass der Prototyp die spezifizierten funktionalen Anforderungen erfüllt und vielversprechende Ansätze in Bezug auf nicht-funktionale Anforderungen aufweist. Damit stellt die Arbeit einen gelungenen und validierten Proof of Concept dar, der die grundsätzliche Machbarkeit einer KI-gestützten natürlichen Sprachschnittstelle für Process Flow belegt.
Aufbauend auf diesen Ergebnissen eröffnen sich konkrete Entwicklungsperspektiven. Für eine produktive Nutzung müsste die Funktionalität auf die vollständige Geschäftslogik des Process Flow-Backends erweitert werden. Der Austausch in natürlicher Sprache könnte durch fortgeschrittene MCP-Features wie Elicitation und Sampling eine deutlich nutzerzentriertere Dialogführung ermöglichen.

A Literaturverzeichnis

[1] A. Gadatsch, *Modellierung und Analyse von Prozessen*, pp. 107–181. Wiesbaden: Springer Fachmedien Wiesbaden, 2025.

[2] T. Keuthen, *Prompt Engineering im Geschäftsprozessmanagement: Innovation mit KI als Schlüssel zur Effizienz*. Wiesbaden: Springer Fachmedien Wiesbaden, 2025.

[3] T. Amaratunga, "Transformers," in *Understanding Large Language Models*, pp. 55–79, United States: Apress L. P, 2023.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[5] S. Balasubramaniam, S. Kadry, A. Prasanth, R. Kumar Dhanaraj, A. Prasanth, R. Kumar Dhanaraj, S. Kadry, and S. Balasubramaniam, *Generative AI and LLMs: Natural Language Processing and Generative Adversarial Networks*. Berlin/Boston: De Gruyter, 1 ed., 2024.

[6] Werkzeugmaschinenlabor der RWTH Aachen, "Abteilung produktionsmanagement." Informationsseite der Abteilung Produktionsmanagement. Abgerufen am 14.12.2025. Adresse: https://www.wzl.rwth-aachen.de/cms/wzl/forschung/produktionssystematik/ suqo/produktionsmanagement/.

[7] Abteilung Produktionsmanagement des Werkzeugmaschinenlabors der RWTH Aachen, "Aixperanto." Informationsseite über Aixperanto. Abgerufen am 14.12.2025. Adresse: http://aixperanto.com/de/default.html.

[8] J. Glück, "Entwicklung und implementierung eines ki-basierten eingabemoduls für das tool process flow am wzl der rwth aachen," Aug. 2025.

[9] Ollama, "Ollama documentation," 2025. Offizielle Dokumentation zu Ollama-API. Abgerufen am 14.12.2025. Adresse: https://docs.ollama.com/api.

[10] Anthropic, "Introduction to the model context protocol," 2025. Offizielle Einführungsdokumentation des Model Context Protocol (MCP). Abgerufen am 09.12.2025. Adresse: https://modelcontextprotocol.io/.

[11] A. Tynamo, “Introducing the model context protocol,” 2024. Offizielle Ankündigung und Motivation. Abgerufen am 09.12.2025. Adresse: https://www.anthropic.com/news/model-context-protocol.

B Abbildungsverzeichnis