

# Schlussbericht

zum Vorhaben



Thema:

**Daten- und Modellbasiertes Informationsmanagement für die Systementwicklung**

Zuwendungsempfänger:

**EXP Software GmbH**

Förderkennzeichen:

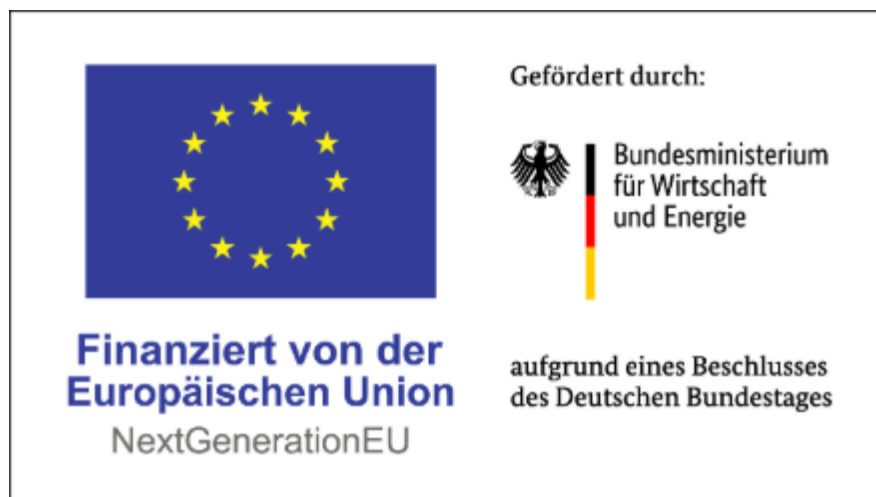
**19S22008B**

Laufzeit:

**von: 01.12.2022 bis: 30.11.2025**

Monat der Erstellung:

**01/2026**



# Inhaltsverzeichnis

<b>I.</b>	<b>Kurzbericht</b> .....	<b>1</b>
1.	Aufgabenstellung.....	1
2.	Voraussetzungen, unter denen das Vorhaben durchgeführt wurde.....	1
3.	Planung und Ablauf des Vorhabens.....	1
4.	Wissenschaftlichem und technischem Stand.....	1
5.	Zusammenarbeit mit anderen Stellen.....	2
<b>II.</b>	<b>Ausführliche Darstellung der Ergebnisse</b> .....	<b>3</b>
1.	Erzielte Ergebnisse.....	3
	Arbeitspaket 1: Engineering Architecture Model.....	3
	Arbeitspaket 2: Datenquellen und IT-Systeme.....	5
	Arbeitspaket 3: Traceability zum Informationsmanagement.....	9
	Arbeitspaket 4: Kontextbasierte Enterprise Search.....	11
	Arbeitspaket 5: Modellbasierte Simulation des Entwicklungsprozesses.....	15
	Arbeitspaket 6: Projektmanagement / Transfer / Datenmanagement.....	17
2.	Wichtigste Positionen des zahlenmäßigen Nachweises.....	18
3.	Notwendigkeiten und Angemessenheiten der geleisteten Arbeiten.....	19

# I. Kurzbericht

## 1. Aufgabenstellung

Das Forschungsprojekt „MoInSe – Daten- und modellbasiertes Informationsmanagement für die Systementwicklung“ verfolgte das Ziel, eine konzeptionelle sowie prototypische Engineering-IT-Plattform zu entwickeln, die den wachsenden Anforderungen bei der Entwicklung vernetzter und softwareintensiver Fahrzeuge gerecht wird. Im Mittelpunkt stand dabei die Verknüpfung heterogener Produktmodelle und der zugehörigen IT-Systeme, um eine anwendungsübergreifende, kontextbezogene und benutzerfreundliche Suche, eine durchgängige Nachverfolgbarkeit der Beziehungen zwischen Entwicklungsartefakten sowie eine datenbasierte Planung zukünftiger Entwicklungsprojekte zu ermöglichen.

Zu diesem Zweck wurden Prozesse, Werkzeuge, Informationsstandards, Organisationsstrukturen und Aktivitäten der Systementwicklung systematisch analysiert. Die relevanten Datenquellen sollten mit unterschiedlicher Integrationstiefe in einem Federation Layer zusammengeführt und ihre Metadaten kontextsensitiv miteinander vernetzt werden. Ergänzend war die Entwicklung methodischer und technischer Unterstützungsansätze zur projektbegleitenden Erzeugung von Tracelinks vorgesehen, um darauf aufbauend eine kontextbasierte Enterprise Search sowie eine Forecast- und Simulations-Engine zu implementieren und diese anhand eines Referenzprojekts zu evaluieren.

## 2. Voraussetzungen, unter denen das Vorhaben durchgeführt wurde

Das Projekt wurde im Rahmen der Fördermaßnahme „Neue Fahrzeug- und Systemtechnologien“ des Bundesministeriums für Wirtschaft und Energie durchgeführt. Die Partner arbeiteten gemeinschaftlich und kooperativ an der Erreichung der Projektziele. Die Mischung aus Wirtschaft und Wissenschaft brachte interdisziplinäre Expertise zusammen, um die für die Erreichung der Projektziele notwendige Innovationskraft zu entfalten. Verantwortlicher Projektträger war die TÜV Rheinland Forschungs- und Innovationsmanagement GmbH.

## 3. Planung und Ablauf des Vorhabens

Das Forschungsprojekt war in sechs Arbeitspakete (AP) gegliedert, die sich an der Zielarchitektur der Engineering-IT-Plattform ausrichteten. Zunächst war der Aufbau der PTIO-Landschaft, des Engineering Architecture Models sowie des Engineering Knowledge Graphs vorgesehen (AP 1). Darauf aufbauend sollten die relevanten Datenquellen und IT-Systeme identifiziert und in einem Federation Layer zusammengeführt werden (AP 2).

Parallel dazu umfasste das Projekt die Konzeption und Implementierung von Traceability-Mechanismen (AP 3), einer kontextbasierten Enterprise Search mit einem LLM-basierten Assistenzsystem (AP 4) sowie eines modellbasierten Forecast-Ansatzes (AP 5). Ergänzend wurden Projektmanagement-, Transfer- und Datenmanagementaktivitäten im Rahmen eines übergreifenden Arbeitspakets umgesetzt (AP 6).

Die Umsetzung erfolgte agil in 34 Sprints mit standardisierten Plannings, Reviews und Retrospektiven sowie einer einheitlichen Projekt- und Datenstruktur. Zentrale Arbeitsschritte wurden in gemeinsamen Workshops abgestimmt, die drei Meilensteine strukturierten den inhaltlichen Fortschritt. Während AP 1 und AP 2 im Wesentlichen planmäßig abgeschlossen werden konnten, führten organisatorische Änderungen bei John Deere sowie die Migration der IT-Infrastruktur zu EXP inklusive Neuaufbau der Kubernetes-Umgebung und compliance-konformer LLM-Anbindung dazu, dass die verfügbaren Ressourcen auf Federation Layer, Traceability und Enterprise Search fokussiert wurden und die Forecast Engine in AP 5 überwiegend auf konzeptioneller Ebene verblieb.

## 4. Wissenschaftlichem und technischem Stand

Zur Durchführung des Forschungsprojekts wurden keine spezifischen Schutzrechte genutzt oder lizenziert. Das Vorhaben baut damit nicht unmittelbar auf bestehenden Patenten oder geschützten Konstruktionen auf. Technisch wurde auf etablierte Plattformen und Open-Source-Komponenten zurückgegriffen, insbesondere auf Cloud-Infrastrukturen (u.a. Microsoft Azure) sowie die Kogito Process Engine für die Modellierung und Ausführung von Geschäftsprozessen. Methodisch

knüpfte das Projekt an bekannte Verfahren der Enterprise-Architecture- und EAM-Modellierung (für AP 1) sowie an gängige ETL-Verfahren zur Extraktion, Transformation und Laden von Daten aus den Mock-Services und angebundenen Quellsystemen an.

Während der Projektlaufzeit kam es zu einem erheblichen Fortschritt im Bereich KI-basierter Methoden, insbesondere bei Large Language Models (LLMs). Diese Entwicklungen wurden kontinuierlich verfolgt, bewertet und – wo fachlich sinnvoll – in Form von LLM-basierten Verfahren für Suche, Wissensabfrage und Tracelink-Erzeugung in das Projekt integriert.

#### **5. Zusammenarbeit mit anderen Stellen**

Alle erforderlichen Arbeiten konnten durch die Konsortialpartner abgedeckt werden. Eine Einbindung weiterer externer Stellen war daher nicht vorgesehen.

## **II. Ausführliche Darstellung der Ergebnisse**

### **1. Erzielte Ergebnisse**

Im Folgenden werden die Ergebnisse detailliert nach Arbeitspaketen geordnet aufgezeigt.

#### **Arbeitspaket 1: Engineering Architecture Model**

##### **AP 1.1: Modellierung der PTIO-Landschaft**

Das Forschungsprojekt MoInSe startete im Dezember 2022 und hat in Absprache mit dem Projektträger die Arbeit im Februar 2023 aufgenommen. Im Rahmen von AP 1.1 unterstützte die EXP methodisch bei der Durchführung der Workshops zur Aufnahme und Modellierung der PTIO-Landschaft (Product Type Information Object). Für die Anforderungserhebung und Prozessmodellierung wurden zwei etablierte Methodiken eingesetzt: Event Storming ist eine kollaborative Modellierungstechnik, bei der Domänenexperten und Entwickler gemeinsam komplexe Geschäftsprozesse durch das Identifizieren von Domänenereignissen (Domain Events) erschließen. Die Methodik ermöglicht es, auch bei komplexen Systemlandschaften schnell ein gemeinsames Verständnis der relevanten Geschäftsvorgänge zu entwickeln. Domain Story Telling ist eine weitere kollaborative Methodik, bei der die Beteiligten Geschäftsprozesse in Form von "Geschichten" erzählen, die dann in einer standardisierten Notation visualisiert werden. Diese Methodik eignet sich besonders gut, um die Interaktionen zwischen verschiedenen Akteuren, Systemen und Arbeitsobjekten zu erfassen. EXP übernahm bei beiden Methodiken die Rolle der Moderation und der methodischen Ausführung. In beiden Fällen war das Ziel, den jeweiligen Teilprozess möglichst umfassend zu erarbeiten, inklusive der beteiligten Systeme, Rollen, Aggregate und Daten. Die Workshops wurden mit Fachexperten (Subject Matter Experts) von John Deere durchgeführt, die tiefes Domänenwissen über die Engineering-Prozesse im Maschinenbau einbrachten.

Aus den Ergebnissen der Workshops entstanden BPMN-Modelle (Business Process Model and Notation), die syntaktisch korrekt und maschinenlesbar die Geschäftsprozesse des PTIO abbilden. Verwendet wurde dabei der reine BPMN 2.0 Standard wie von der Object Management Group (OMG) definiert, um eine maximale Kompatibilität mit etablierten Werkzeugen wie Modelio und Camunda zu erreichen. Diese Entscheidung war strategisch wichtig, da sie die spätere Ausführbarkeit der Prozesse in einer Process Engine sicherstellte. Die BPMN-Modelle umfassten unter anderem den Produktkonfigurationsprozess, der den kompletten Workflow von der Konfigurationsänderung über die Definition, Prüfung und Freigabe bis zur Implementierung abbildet. Dieser Prozess involviert verschiedene Rollen wie den Configuration Engineer, den Konstrukteur und den Engineering Analyst sowie Systeme wie das Change Management System (CMS) und das Product Lifecycle Management System (PLM).

Die abgeleiteten BPMN-Modelle wurden nach Feedback der Subject Matter Experts mehrfach verfeinert und revidiert. Mit Blick auf die Anforderungen von AP 2 (Datenquellen und IT-Systeme) wurden die Modelle durch die beteiligten Systeme ergänzt, dargestellt als Blackbox Pools gemäß dem BPMN-Standard. Diese Darstellungsform erlaubt es, externe Systeme als Kommunikationspartner abzubilden, ohne deren interne Abläufe im Detail modellieren zu müssen. Die beiden Methodiken Event Storming und Domain Story Telling wurden evaluiert, verglichen und anhand der Qualität der Ergebnisse bewertet. Diese Evaluation lieferte wertvolle Erkenntnisse darüber, welche Methodik für welche Art von Prozessen besonders geeignet ist und wie beide Ansätze komplementär eingesetzt werden können. Sämtliche BPMN-Dateien wurden im projekteigenen Git-Repository revidiert. Dies ermöglichte eine vollständige Nachvollziehbarkeit aller Änderungen, die kollaborative Bearbeitung durch verschiedene Projektpartner sowie die Integration in die CI/CD-Prozesse des Projekts.

##### **AP 1.2: Analyse der Verknüpfungen der PTIO-Sichten**

Im weiteren Projektverlauf unterstützte EXP mit technischem Input zu den Modellierungen. Dies betraf insbesondere die Auswahl einer passenden Modellierungssprache und die Unterstützung bei der korrekten Methodik. Der Schwerpunkt lag dabei auf der Ausführbarkeit und Nutzbarkeit der erarbeiteten Modelle in den späteren Implementierungsphasen. Die EXP stellte sicher, dass die Modelle nicht nur dokumentarischen Wert hatten, sondern tatsächlich als Grundlage für die technische Implementierung dienen konnten. Die konsequente Versionierung der Arbeit in Git war ein zentraler Aspekt der Qualitätssicherung. Durch die Verwendung von Git als Versionskontrollsystem konnten alle Änderungen an den Modellen nachvollzogen, verschiedene Versionen verglichen und bei Bedarf auf frühere Stände zurückgegriffen werden.

EXP unterstützte das Projekt durch die Betreuung einer angeschlossenen Bachelorarbeit an der Technischen Hochschule Ingolstadt. Die Arbeit mit dem Titel „Entwicklung und Evaluation eines Vorgehensmodells für die Entwicklung von Geschäftsprozessen im Engineering“ wurde von Philipp Pfaller verfasst und behandelte die methodischen Grundlagen der Geschäftsprozessmodellierung im Engineering-Kontext.

EXP beriet hinsichtlich der korrekten Aufarbeitung der Modelle, um einen möglichst hohen Mehrwert für die Implementierung sicherzustellen. Die Erkenntnisse aus der Bachelorarbeit flossen direkt in die Projektarbeit ein und trugen zur methodischen Fundierung des Vorgehens bei.

Die von dem Projektpartner Fraunhofer IEM entwickelten BPMN-Prozesse wurden von EXP einer eingehenden Überprüfung und einem umfassenden Review unterzogen. Im Rahmen dieser Überprüfung wurde sowohl semantisches als auch syntaktisches Feedback vom gesamten MoInSe-Team eingeholt. Die Ausführbarkeit der BPMN-Prozesse wurde systematisch verifiziert, um sicherzustellen, dass die Prozesse später in einer Process Engine wie Kogito ausgeführt werden können. EXP korrigierte die identifizierten Problemstellen. Einige Prozesse wurden zudem im Hinblick auf BPMN-Best Practices optimiert, um ihre Lesbarkeit und Verständlichkeit zu erhöhen

Das Konzept der Traceability aus AP 3 wurde in die PTIO-Landschaft integriert. Diese Integration war wichtig, um die Verbindungen zwischen den modellierten Prozessen und den später zu implementierenden Tracelinks im Knowledge Graph herzustellen. Durch die frühzeitige Integration des Traceability-Konzepts konnte sichergestellt werden, dass die Prozessmodelle alle notwendigen Informationen enthalten, um später die Nachverfolgbarkeit von Änderungen und Abhängigkeiten zu ermöglichen.

Die von John Deere erstellte Beispielfragenliste wurde auf technische Machbarkeit hin überprüft. Diese Fragenliste enthielt typische Anfragen, die Ingenieure und Projektmanager im Alltag an ein wissensbasiertes System stellen würden.

Da als Eingangsdaten ein nicht produktives Beispielprojekt (der sogenannte "Toy Tractor") vorlag, war es besonders wichtig, jede Frage anhand der verfügbaren Eingangsdaten zu analysieren und entsprechendes Feedback zu geben. Hierbei wurde ein Feedback-Loop gestartet, um kontinuierlich bessere Beispiel- und Testdaten zu generieren. Diese iterative Vorgehensweise stellte sicher, dass die Modellierung praxisnah und anwendungsorientiert erfolgte. Erste Konzepte für die Notwendigkeit einer Testsuite für die Antworten sind entstanden, welche später (s.u.) in die Implementierung von Rage4j mündeten.

### **AP 1.3: Knowledge-Graph-Datenmodell**

Die Entwicklung des Knowledge-Graph-Datenmodells erfolgte in enger Zusammenarbeit zwischen EXP, dem IEM und John Deere. In mehreren iterativen Workshops wurden die Anforderungen aus den verschiedenen Domänen (CAD, CAM, PLM, PDM, ERP, Organisation/Personen, Prozesse) in ein konsistentes Datenmodell überführt. Die in AP 1.1 und AP 1.2 gewonnenen Erkenntnisse aus der PTIO-Modellierung und der Analyse der Sichten bildeten dabei die fachliche Grundlage. Ein zentrales Ergebnis dieser Zusammenarbeit war die Definition eines stark typisierten Graphmodells, welches die heterogenen Datenstrukturen der verschiedenen Engineering-Systeme in einer einheitlichen Struktur abbildet. Das Modell berücksichtigt sowohl die Knotentypen (z. B. Produkt, Komponente, Prozessschritt, Person, Dokument) als auch die Beziehungstypen zwischen diesen Entitäten.

Ein wesentlicher Aspekt der Aufgabenstellung war die Reflexion des Datenmodells gegen die tatsächliche technische Umsetzung. Bei der Evaluation verschiedener Graph-Datenbanken (neo4j, OrientDB, ArcadeDB, ArangoDB, GraphDB) mittels der Pugh-Methode erwies sich die Unterstützung von Strong Typing als entscheidendes Kriterium. Die Entscheidung fiel auf ArcadeDB, da diese Datenbank im Gegensatz zu anderen Lösungen ein echtes Typsystem unterstützt, das die definierten Knotentypen und Beziehungstypen als Schema in der Datenbank abbilden kann. Diese technische Eigenschaft ermöglichte es, das gemeinsam mit IEM und John Deere entwickelte konzeptionelle Datenmodell direkt als Typsystem in der Graph-Datenbank zu implementieren. Dadurch werden Datenintegrität und Konsistenz auf Datenbankebene sichergestellt – fehlerhafte oder nicht dem Schema entsprechende Daten werden bereits beim Einfügen abgelehnt.

Das implementierte Datenmodell wurde kontinuierlich gegen die in AP 1.2 definierten Anwendungsfälle und Sichten validiert. Die aus dem Event Storming und Domain Story Telling gewonnenen Szenarien dienten als Testfälle, um sicherzustellen, dass alle relevanten Abfragen und Traversierungen durch das Graphmodell effizient unterstützt werden. Besonders die Integration der Traceability-Anforderungen –

also die Nachverfolgbarkeit von Änderungen und Abhängigkeiten über verschiedene Domänengrenzen hinweg – wurde als wichtiger Validierungsaspekt identifiziert. Das typisierte Graphmodell erwies sich hier als besonders geeignet, da Beziehungen zwischen Entitäten explizit modelliert und traversiert werden können. Im Rahmen der kontinuierlichen Reflexion gegen die technische Implementierung entstand die Hypothese, dass ein stark typisiertes Graphmodell besonders vorteilhaft für den Einsatz von KI-Methoden ist. Die explizite Schemastruktur bietet klare Anknüpfungspunkte für Retrieval Augmented Generation (RAG) und ermöglicht präzisere Abfragen als schemalose Ansätze. Diese Erkenntnis floss in die Planung der nachfolgenden Arbeitspakete zur KI-Integration ein.

## **Arbeitspaket 2: Datenquellen und IT-Systeme**

### **AP 2.1: Technische Evaluierung der IT-Infrastruktur**

Die technische Evaluierung der IT-Infrastruktur begann mit einer systematischen Analyse verschiedener Graph-Datenbanken. In einer Kubernetes-Umgebung des Partners John Deere (CaaS – Container as a Service) wurden mittels Helm-Charts die Datenbanksysteme neo4j, OrientDB, ArcadeDB, ArangoDB und GraphDB installiert und mit einem einheitlichen Testdatensatz ausgestattet. Die Evaluation erfolgte nach der Pugh-Methode, wobei technische Eigenschaften wie Skalierbarkeit, Hochverfügbarkeit und die Qualität der Schnittstellen bewertet wurden. Nach Abschluss der Evaluation fiel die Entscheidung auf ArcadeDB als zentrale Datenhaltung für den Knowledge Graph. ArcadeDB überzeugte insbesondere durch seine starke Typisierung, bei der Knoten (Vertizes) und Kanten (Edges) als Instanzen von Klassen behandelt werden können. Diese Klassen können wiederum Subklassen anderer Klassen sein, wodurch sich Konzepte der Objektorientierung elegant auf die Graphenstruktur übertragen lassen. Für die Behandlung von Semantik und Ontologie im Engineering-Kontext stellte dies einen entscheidenden Qualitätsfaktor dar. Ferner bietet ArcadeDB native Multi-Model-Unterstützung für Graph-, Dokumenten-, Key-Value- und Time-Series-Daten in einer einzigen Datenbank, was die Flexibilität der späteren Datenmodellierung erheblich erhöht.

Ein zentrales Element der technischen Infrastruktur bildet der verteilte Event-Bus auf Basis von Apache Kafka. Kafka dient in der MoInSe-Architektur als entkoppelnde Schicht zwischen den Importer-Services und dem Knowledge Graph. Durch diese Entkopplung ist das System nicht auf synchrone API-Aufrufe angewiesen und kann alle Resilienz-Eigenschaften von Kafka nutzen, darunter garantierte Nachrichtenzustellung, Fehlertoleranz und horizontale Skalierbarkeit. Da im Projektverlauf potenziell große Datenmengen im Big-Data-Bereich zu verarbeiten sind, wurde der Kafka-Cluster von EXP verteilt ausgelegt. Der Cluster ist dediziert für MoInSe implementiert und von anderen Projekten isoliert. EXP verantwortete die Auslegung, Installation und Implementierung der CI/CD-Pipeline für den Kafka-Cluster sowie die fortlaufende Wartung und Updates. Im weiteren Projektverlauf wurde zusätzlich eine Schema-Registry auf Basis von Confluent eingeführt, um die Datenqualität durch Avro-Schemata abzusichern. Diese Schema-Validierung stellt sicher, dass nur korrekt strukturierte Daten in den Knowledge Graph gelangen und dessen Qualität aufrechterhalten wird.

Als technologische Basis für die Backend-Services wurde das Quarkus-Framework gewählt. Quarkus ist ein Kubernetes-natives Java-Framework, das speziell für Cloud-Umgebungen und containerisierte Deployments optimiert ist. Die Vorteile von Quarkus für das MoInSe-Projekt liegen in den kurzen Startzeiten, dem geringen Speicherverbrauch und der nahtlosen Integration mit Cloud-Native-Technologien. Alle von EXP entwickelten Microservices basieren auf Quarkus und nutzen dessen reaktive Programmiermodelle für eine effiziente Verarbeitung der Datenströme. Die Service-Architektur folgt dem Prinzip der Trennung von Fachlogik und Datenimport. Um im Federation Layer nicht für jede neue Schnittstelle den kompletten Stack neu entwickeln zu müssen, wurde eine modulare Architektur erarbeitet. Diese Architektur trennt die domänenspezifische Fachlogik von der generischen Logik zum Importieren der Graphenstrukturen. Die einzelnen Services kommunizieren über definierte Schnittstellen und den zentralen Kafka-Event-Bus, wodurch eine lose Kopplung und hohe Wartbarkeit gewährleistet werden. Die technische Evaluation der anzubindenden Quellsysteme konzentrierte sich exemplarisch auf zwei Systeme, die unterschiedliche Extremata im Spektrum der Datenstrukturierung repräsentieren. Das PLM-System Windchill von PTC verfügt über ein stark strukturiertes Datenmodell und stellt Daten über eine RESTful API bereit. Diese selbstbeschreibende API-Architektur ermöglicht eine flexible Navigation durch die Ressourcen und erleichtert die Extraktion von BOM-Strukturen (Bill of Material) und CAD-Referenzen. Als Gegenpol wurde Microsoft SharePoint evaluiert, das primär unstrukturierte Dokumenteninhalte verwaltet.

Die Absicherung der gesamten Infrastruktur erfolgte über standardisierte Protokolle. Alle Services wurden mittels OAuth2 und OpenID Connect (OIDC) abgesichert. Initial kam OKTA als Identity Provider

zum Einsatz, um eine sichere tokenbasierte Authentifizierung zu gewährleisten. Als zentraler Einstiegspunkt für alle API-Anfragen wurde ein APISIX-Gateway genutzt.

Alle Softwarekomponenten wurden auf dem Kubernetes-Cluster in der CaaS-Umgebung von John Deere deployt. Für die kontinuierliche Integration und Auslieferung implementierte EXP eine CI/CD-Pipeline auf Basis von GitHub Actions. Die Helm-Charts für alle Services wurden in einem zentralen GitHub-Repository versioniert und sind für das gesamte Konsortium einsehbar. Die Pipeline umfasst automatisierte Tests, Container-Image-Builds und das Deployment in die verschiedenen Umgebungen. EXP übernahm dabei nicht nur das Deployment der eigenen Services, sondern auch die Auslieferung der Softwarekomponenten der anderen Projektpartner.

## **AP 2.2: Analyse der Integrationstiefe**

Die Analyse der Integrationstiefe führte zur Entwicklung eines innovativen Architekturkonzepts, das sowohl hohe Modularität als auch strikte Einhaltung der Datensicherheitsanforderungen gewährleistet. Anstatt Daten direkt aus den produktiven Quellsystemen von John Deere zu beziehen, wurde ein Konzept sogenannter Mock Services erarbeitet. Diese Services verhalten sich nach außen wie die Ursprungssysteme, beinhalten jedoch ausschließlich die für das Forschungsprojekt benötigten Daten. Durch diese strikte Trennung von Demonstrations- und Produktivdaten werden die Vorgaben an Datensicherheit und Datenschutz konsequent eingehalten. Insbesondere werden personenbezogene Daten durch synthetische Testdaten ersetzt, bevor sie in die MolnSe-Infrastruktur gelangen.

Für die Anonymisierung der Organisationsdaten aus dem Enterprise Datalake entwickelte EXP nach umfassender Recherche der rechtlichen Hintergründe und mathematischen Grundlagen einen Anonymisierungsalgorithmus. Dieser Algorithmus verarbeitet die Originaldaten und stellt sie anschließend über eine EDL-ähnliche Schnittstelle bereit, sodass die nachgelagerten Importer-Services keine Anpassungen an ihrer Datenverarbeitung vornehmen müssen.

Die Mock-Services werden von jeweils dedizierten Importer-Services abgefragt, die die Daten bei Bedarf transformieren und anreichern. Ein zentrales Beispiel hierfür ist der s.g. D12N-Importer-Service, der aus den einzelnen Konstruktionsdaten des Windchill-Mock-Services eine hierarchische Baumstruktur der Baukarte (Bill of Material) aufbaut. Diese Baumstruktur ist mathematisch betrachtet ein Graph ohne Zyklen und kann daher direkt in den Knowledge Graph übernommen werden. Die transformierten Daten werden anschließend auf den verteilten Apache Kafka Event-Bus geschrieben. Im weiteren Projektverlauf wurde die Datenqualität durch die Einführung von Avro-Schemata und einer Confluent Schema Registry abgesichert. Avro als kompaktes Binärformat ermöglicht eine effiziente Serialisierung der Daten, während die Schema Registry sicherstellt, dass Produzenten und Konsumenten stets kompatible Schemata verwenden. Ein dedizierter ArcadeDB Importer Service liest die Events vom Kafka-Bus und persistiert sie im Knowledge Graph. Für die Speicherung von Binärdaten wie 3D-CAD-Dateien wurde MinIO als S3-kompatibler Objektspeicher in die Architektur integriert. Bevor die Metadaten der CAD-Objekte auf Kafka abgelegt werden, werden die zugehörigen 3D-Dateien auf den MinIO-Server hochgeladen. Im Knowledge Graph wird anschließend lediglich der Link auf diese Dateien gespeichert, wodurch die Graph-Datenbank von großen Binärdaten entlastet wird und eine klare Trennung von Metadaten und Nutzdaten erreicht wird.

Ein wesentlicher Aspekt der Integrationstiefe betrifft die Rollen- und Rechteverwaltung. Hierfür wurde ein Konzept erarbeitet, das die Organisationsstruktur von John Deere als Graph abbildet und diese Informationen für eine feingranulare Zugriffskontrolle nutzt. Das von EXP konzipierte Org-Tool baut aus den anonymisierten EDL-Daten eine Graphstruktur der Organisation auf und stellt Schnittstellen zur Einpflege von Daten über Abteilungen, Arbeitsgruppen und Rollen bereit. Für die eigentliche Zugriffskontrolle wurde SpiceDB als Lösung für Fine-Grained Authorization (FGA) ausgewählt. SpiceDB implementiert das von Google entwickelte Zanzibar-Modell für Berechtigungsprüfungen und ermöglicht es, komplexe Beziehungen zwischen Benutzern, Gruppen und Ressourcen effizient abzufragen. Die Organisationsdaten werden sowohl in der ArcadeDB als Teil des Knowledge Graphs als auch in SpiceDB für die Berechtigungsprüfung vorgehalten. Diese Trennung der Belange (Separation of Concerns) stellt sicher, dass der Knowledge Graph nicht selbst über Zugriffsberechtigungen entscheiden muss. Mithilfe von OIDC-Token wird der Benutzer identifiziert und die mit ihm verknüpften Berechtigungen aus SpiceDB abgefragt. EXP übernahm die Installation von SpiceDB, die Implementierung der CI/CD-Pipeline sowie die Synchronisation der relevanten Teilgraphen zwischen ArcadeDB und SpiceDB. Später wurde dann von SpiceDB hin zu OpenFGA gewechselt, welches denselben Zweck erfüllt.

Um die in AP 1 modellierten Geschäftsprozesse nicht nur statisch abzubilden, sondern auch deren Ausführung zu erfassen, wurde eine Process Engine in die Architektur integriert. Das Ziel war es, ein

lebendiges Modell zu schaffen, das Prozessinstanzen, deren Status und Verlauf im Knowledge Graph abbildet.

Die Auswahl der geeigneten Process Engine war von technischen Herausforderungen geprägt. Erste Implementierungsversuche erfolgten mit Flowable Open Source, einer weit verbreiteten BPMN-Engine. Im Verlauf der Integration zeigten sich jedoch erhebliche Probleme: Die Dokumentation der Open-Source-Version war mangelhaft und in neueren Versionen wurden wesentliche Features entfernt oder hinter einer kommerziellen Lizenz versteckt. Nach eingehender Evaluation wurde daher die Entscheidung getroffen, auf Kogito umzusteigen.

Kogito ist eine cloudnative Open Source Workflow-Engine, die von Red Hat als Teil des Quarkus-Ökosystems entwickelt wird. Kogito ist funktional sehr umfangreich und lässt sich nahtlos in die bestehende Quarkus-basierte Microservice-Architektur integrieren. Ferner bietet Kogito ein integriertes User Interface für die Prozessvisualisierung und -steuerung, das von EXP exemplarisch in einer lokalen Entwicklungsumgebung implementiert wurde. Während der Projektlaufzeit wurde Kogito von Red Hat an die Apache Software Foundation übertragen.

### **AP 2.3: Aufbau und Anbindung der IT-Systeme an den Federation Layer**

Der Aufbau des Federation Layers umfasste die Entwicklung und das Deployment zahlreicher Microservices, die gemeinsam die Brücke zwischen den Quellsystemen und dem Knowledge Graph bilden. EXP implementierte alle Mock Services für die verschiedenen Quellsysteme sowie den generischen Importer Service, der als Template für domänenspezifische Importlogik dient. Die Services wurden als containerisierte Anwendungen entwickelt und folgen den Prinzipien einer Cloud-nativen Architektur mit Health Checks, Metrics-Endpoints und konfigurierbaren Umgebungsvariablen.

Der D12N Importer Service bildet das Herzstück der Windchill-Integration. Er ruft die CAD-Daten vom Windchill Mock Service ab, transformiert die flachen Datensätze in eine hierarchische BOM-Struktur und verlinkt die einzelnen Elemente mit den zugehörigen 3D-Daten auf dem MinIO-Server. Die resultierenden Graphstrukturen werden als Avro-serialisierte Events auf Kafka publiziert, von wo sie der ArcadeDB Importer Service konsumiert und in den Knowledge Graph schreibt.

Für die Integration des Organisationsdiagramms wurde der EDL Mock Service entwickelt, der anonymisierte Organisationsdaten über eine JDBC-kompatible Schnittstelle bereitstellt. Das Org-Tool verarbeitet diese Daten und generiert daraus eine navigierbare Graphstruktur mit Abteilungen, Teams, Rollen und deren Beziehungen untereinander.

Im Verlauf des Projekts wurde das Konzept des Event Sourcings vertieft, um nicht nur den aktuellen Zustand, sondern auch die Historie von Änderungen im Knowledge Graph abzubilden. In Zusammenarbeit mit einer angeschlossenen Masterarbeit wurde das Konzept des Temporal Property Graphs erforscht und implementiert. Dabei werden Änderungen an Knoten und Kanten als zeitgestempelte Events erfasst, wodurch Abfragen wie "Wie sah die BOM-Struktur zu einem bestimmten Zeitpunkt aus?" beantwortet werden können. Die Schema-Verwaltung für die Kafka-Events wurde durch die Integration der Confluent Schema Registry professionalisiert. Alle Avro-Schemata werden zentral registriert und versioniert. Bei Schema-Änderungen prüft die Registry automatisch die Kompatibilität mit bestehenden Versionen und verhindert inkompatible Änderungen, die zu Datenverlusten führen könnten. Ergänzend wurde OpenFGA als Alternative zu SpiceDB evaluiert, um die Flexibilität bei der Wahl der Authorization-Lösung zu erhöhen.

Mit dem Umstieg auf OpenFGA zeigte sich jedoch das Problem fehlender Tooling-Unterstützung. Deswegen wurde eine Admin-UI für OpenFGA entwickelt. Sie basiert auf einem Quarkus-Service, der nicht öffentlich zugänglich ist und ein User-Interface mittels htmx ausliefert.

Die Lösung ist so konzipiert, dass sie von Systemadministratoren einfach genutzt werden kann und Rechte- sowie Rollenzugriffe einzelner Personen ohne großen Aufwand visualisiert. Dadurch lassen sich Probleme in den Ursprungsdaten und im Schema schnell identifizieren.

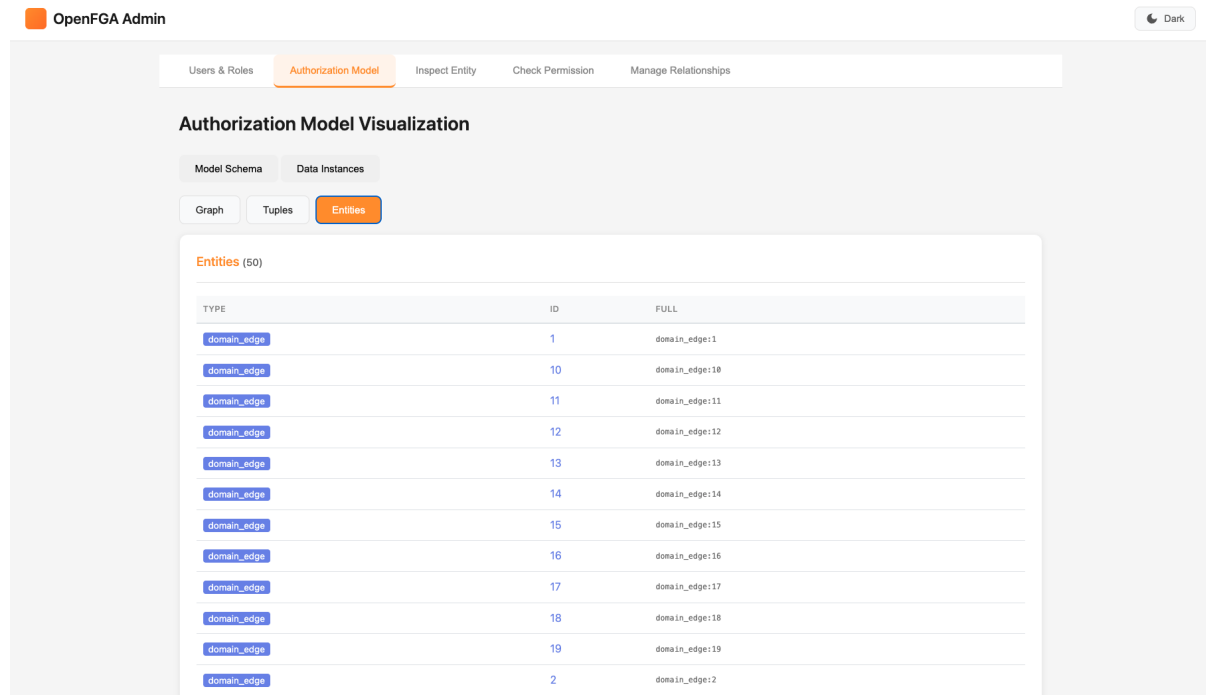


Abb. 1 Admin UI für OpenFGA

Ein einschneidendes Ereignis im Projektverlauf war die Migration der gesamten IT-Infrastruktur von der John Deere CaaS-Umgebung auf Amazon Web Services (AWS). Die ursprüngliche Kubernetes-Umgebung bei John Deere wurde durch einen AWS Elastic Kubernetes Service (EKS) Cluster ersetzt. Diese Migration erforderte umfangreiche Anpassungen an der Deployment-Pipeline und den Konfigurationen aller Services. Im Zuge der Migration wurden auch wesentliche Infrastrukturkomponenten ausgetauscht. Der Identity Provider wurde von OKTA auf Keycloak umgestellt, eine Open-Source-Lösung für Identity und Access Management, die mehr Flexibilität in der Konfiguration und keine laufenden Lizenzkosten verursacht. Die CI/CD-Pipeline wurde von GitHub Actions auf Jenkins migriert, da Jenkins in der neuen Umgebung bereits etabliert war und eine bessere Integration mit den internen Prozessen ermöglichte. Für die Versionsverwaltung wurde ein Wechsel von GitHub zu GitLab vollzogen. Die Deployment-Strategie wurde auf GitOps mit Flux umgestellt. Bei diesem Ansatz wird der gewünschte Zustand der Infrastruktur deklarativ in Git-Repositories definiert. Flux überwacht diese Repositories kontinuierlich und synchronisiert automatisch Änderungen in den Kubernetes-Cluster. Dies erhöht die Nachvollziehbarkeit von Deployments und ermöglicht ein einfaches Rollback auf vorherige Zustände.

In diesem AP integrierte EXP zusätzlich zur ArcadeDB die Graph-Datenbank MicroRelate in die Architektur. MicroRelate ist eine von Grund auf durch EXP entwickelte Komponente und bietet spezifische Vorteile für bestimmte Abfragemuster und ergänzt die Fähigkeiten von ArcadeDB. Für die Abfrage von MicroRelate wurde ein Cypher-Parser implementiert, der die weit verbreitete Graph-Abfragesprache Cypher in die native Abfragesprache von MicroRelate übersetzt. Dies ermöglicht es Entwicklern, mit einer einheitlichen Abfragesprache zu arbeiten, unabhängig davon, welche Graph-Datenbank im Hintergrund angesprochen wird.

Für das Monitoring der Service-Landschaft wurden Grafana und Prometheus in die Infrastruktur integriert. Prometheus sammelt Metriken von allen Services über deren Metrics-Endpoints, während Grafana diese Daten in übersichtlichen Dashboards visualisiert. Die Dashboards ermöglichen es dem Entwicklungsteam, die Systemgesundheit zu überwachen, Performance-Engpässe zu identifizieren und bei Problemen schnell zu reagieren. Die Code-Qualität wird durch SonarQube sichergestellt, das

statische Code-Analysen durchführt und potenzielle Fehler, Sicherheitslücken und Code Smells identifiziert. Für die Messung der Testabdeckung kommt JaCoCo zum Einsatz. Zusätzlich wurde CycloneDX für die Generierung von Software Bill of Materials (SBOM) integriert, um die verwendeten Abhängigkeiten transparent zu dokumentieren und potenzielle Sicherheitsrisiken in Drittbibliotheken frühzeitig zu erkennen.

Mit dem Abschluss von AP 2 wurde eine vollständige, produktionsreife Infrastruktur für den Federation Layer geschaffen. Die entwickelte Architektur ermöglicht die flexible Anbindung heterogener Quellsysteme, die zuverlässige Transformation und den Transport von Daten über Apache Kafka sowie die persistente Speicherung im Knowledge Graph auf Basis von ArcadeDB und MicroRelate. Die Migration auf AWS EKS und die Einführung von GitOps-Praktiken (Flux) haben die Betriebsstabilität und Wartbarkeit des Gesamtsystems deutlich verbessert. Alle Komponenten sind containerisiert, automatisiert getestet und über CI/CD-Pipelines deploybar, wodurch eine solide Grundlage für die nachfolgenden Arbeitspakete geschaffen wurde.

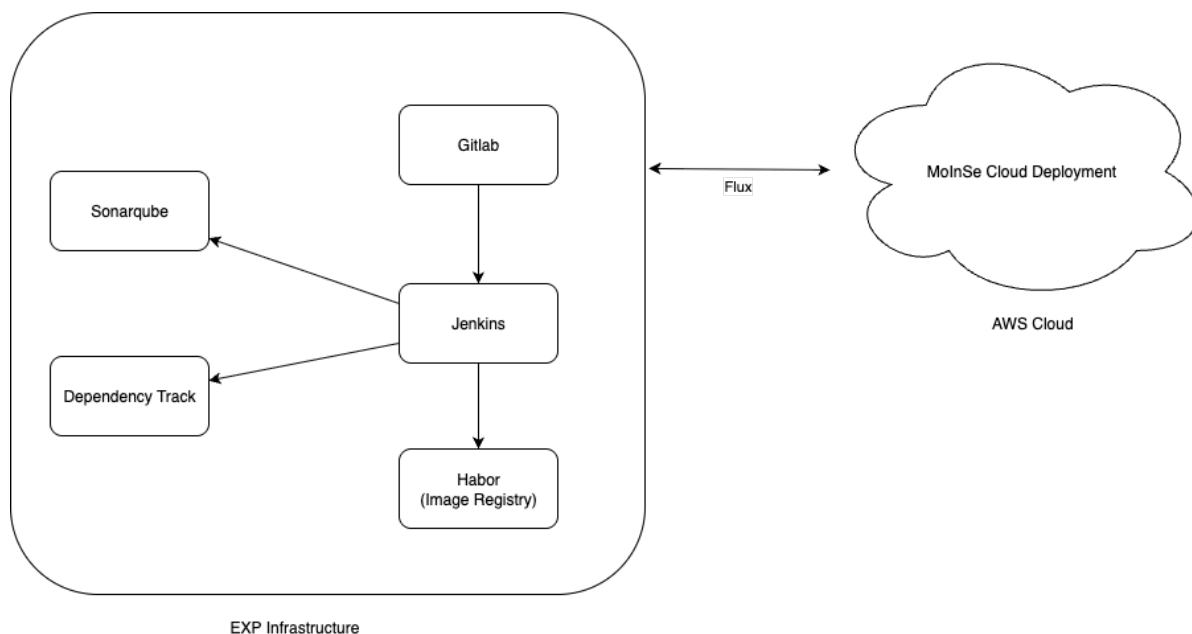


Abb. 2 Neue IT Infrastruktur und CI/CD Pipeline

### Arbeitspaket 3: Traceability zum Informationsmanagement

#### AP 3.1: Evaluierung bestehender Traceability-Methoden

Im Rahmen dieses Arbeitspakets wurde ein umfassendes Konzept zur Traceability im Informationsmanagement entwickelt, das auf der Nutzung von Tracelinks als zentralem Verbindungselement basiert. Tracelinks dienen dazu, Entitäten aus verschiedenen Datenquellen miteinander zu verbinden und ihre Nachverfolgbarkeit über Systemgrenzen hinweg sicherzustellen. Diese Verbindungen bilden das Rückgrat des Knowledge-Graphen und ermöglichen es, Zusammenhänge zwischen CAD-Objekten, Anforderungen, Organisationseinheiten, Prozessen und weiteren Engineering-Artefakten herzustellen.

Die grundlegende Herausforderung bei der Traceability besteht darin, dass die relevanten Verbindungen zwischen Entitäten typischerweise nicht explizit in den Quellsystemen vorhanden sind. Während ein PLM-System die Struktur einer Baukarte kennt und ein Organisationssystem die Hierarchie von Abteilungen abbildet, existieren die semantischen Querverbindungen zwischen diesen Domänen meist nur implizit. Das entwickelte Konzept adressiert diese Herausforderung durch eine Kombination aus regelbasierten und KI-gestützten Ansätzen zur Identifikation und Erstellung von Tracelinks. Für die semantische Klassifikation der Tracelinks wurde ein Typensystem entwickelt, das verschiedene Arten von Beziehungen abbildet. Die Semantik „Identifies“ beschreibt dabei eine eindeutige Zuordnung zwischen Entitäten, etwa wenn ein CAD-Objekt einem bestimmten Part in der Baukarte entspricht. Die Semantik „Relates“ kennzeichnet allgemeine Zusammenhänge ohne strikte Eindeutigkeit, während „Aggregates“ hierarchische Beziehungen wie die Zugehörigkeit eines Bauteils zu einer Baugruppe beschreibt. Diese Typisierung ermöglicht es, bei späteren Abfragen gezielt nach

bestimmten Beziehungsarten zu filtern und die Ergebnisse entsprechend ihrer Semantik zu interpretieren. Das entwickelte Traceability-Konzept wurde in Form eines UML-Diagramms dokumentiert und exemplarisch verifiziert. Die abstrakte Modellierung unterscheidet zwischen Class-Tracelinks, die Beziehungen zwischen Klassen definieren, Object-Tracelinks für konkrete Instanzbeziehungen und Attribute-Tracelinks für Verknüpfungen auf Attributebene. Jeder Tracelink verfügt über einen Score-Wert, der die Konfidenz der Verbindung angibt, sowie Zeitstempel für Erstellung und eventuelle Löschung, um die Historisierung zu unterstützen.

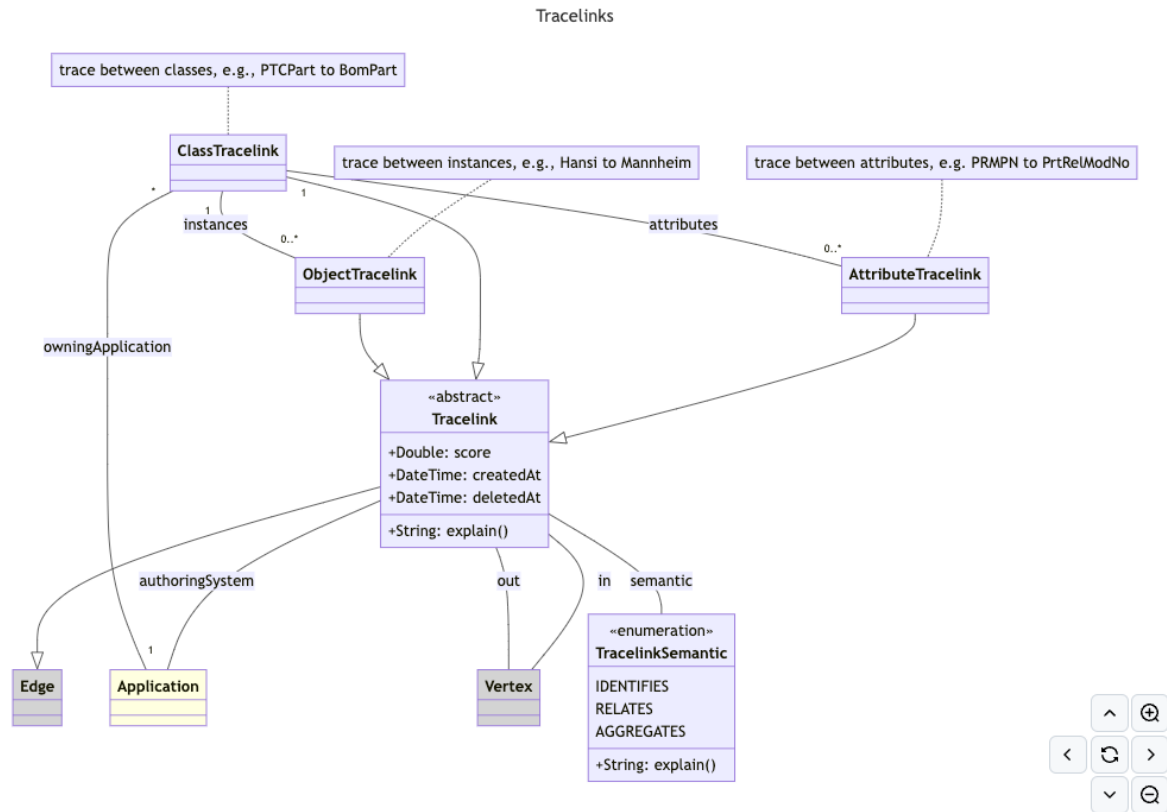


Abb. 3 Beispiele für Semantiken

### AP 3.2: Teilautomatisierte Vernetzung

Die Regeln zur Erstellung von Tracelinks wurden zunächst manuell definiert und in deterministische Algorithmen überführt. EXP konzipierte und implementierte erste regelbasierte Algorithmen, die in die Importer Services integriert wurden. Diese deterministischen Verfahren erkennen Tracelinks anhand von Musterabgleichen und definierten Kriterien, beispielsweise durch den Vergleich von Identifikatoren oder die Analyse von Namenskonventionen.

Als Baseline und Vergleichsgrundlage für fortgeschrittenere Methoden wurde eine traditionelle Suche auf Basis des Page-Rank-Algorithmus entwickelt. Dieser aus dem Web-Kontext bekannte Algorithmus bewertet die Relevanz von Knoten im Graphen anhand ihrer Vernetzung und wird mit einem Breadth-First-Search-Algorithmus kombiniert. Die Implementierung basiert direkt auf der Graph-Struktur des Knowledge Graphs und dient einerseits als deterministischer Algorithmus beim Erstellen von Tracelinks und andererseits als Referenz für die Evaluation neuerer, LLM-basierter Ansätze.

Im weiteren Projektverlauf wurden LLM-gestützte Methoden zur Tracelink-Generierung untersucht. Die ursprüngliche Intention bestand darin, Large Language Models zu nutzen, um paarweise die Ähnlichkeit von Knoten im Knowledge Graph zu bewerten. Diese semantische Ähnlichkeitsbewertung sollte über rein syntaktische Vergleiche hinausgehen und auch implizite Zusammenhänge erkennen können, die für menschliche Experten offensichtlich wären, sich aber nicht durch einfache Regeln formalisieren lassen.

Für die Ausführung der Tracelink-Algorithmen wurde eine serverlose Architektur auf Basis von Knative evaluiert und implementiert. Knative ist eine Kubernetes-native Plattform für serverlose Workloads, die eine bedarfsgesteuerte und ressourceneffiziente Verarbeitung ermöglicht. Im Gegensatz zu AWS Lambda, das nur innerhalb des AWS-Ökosystems verfügbar ist, kann Knative in jedem Kubernetes-Cluster betrieben werden. Nach einer Evaluation verschiedener Alternativen wie KEDA und Fission fiel

die Entscheidung auf Knative aufgrund der umfangreichen Features und der nativen Unterstützung durch das Quarkus-Framework.

Die entwickelte Architektur sieht vor, dass ein Tracelink Consumer eingehende Nachrichten auf dem Kafka-Bus überwacht und bei bestimmten Events dynamisch den zugehörigen Tracelink Service anstößt. Durch das Scale-to-Zero-Prinzip werden Services nur bei Bedarf hochgefahren und anschließend wieder herunterskaliert, was eine effiziente Ressourcennutzung gewährleistet. Jedes individuelle Lambda kann dann spezifische Algorithmen oder KI-Modelle anwenden, beispielsweise zum Extrahieren von Domänen und Fachbereichen aus Anforderungstexten oder zum Verbinden von Engineering Change Requests mit betroffenen Bauteilen. EXP hat mehrere exemplarische Lambdas implementiert, installiert und erprobt sowie die dafür notwendige Infrastruktur implementiert und betrieben.

### **AP 3.3: Integration des Traceability-Netzwerks in den Federation Layer**

Die Integration des Traceability-Netzwerks in den Federation Layer erforderte die Entwicklung einer flexiblen Abstraktionsschicht, die unabhängig von der konkret verwendeten Graph-Datenbank arbeitet. Um die Validierung des Prototyps unabhängig von der zugrundeliegenden Persistenzschicht zu ermöglichen und das Risiko eines Ausfalls bei der Weiterentwicklung einzelner Datenbanken zu reduzieren, wurde der bestehende Cartridge Loader um die Fähigkeit erweitert, neben der ArcadeDB auch die von EXP entwickelte Graph-Datenbank MicroRelate zu versorgen. Die Architektur des Cartridge Loaders wurde dafür modularisiert, sodass ein zusätzlicher Output-Channel konfiguriert werden kann. Innerhalb des Event-basierten Verarbeitungsmodells fungiert der Loader nun als Multi-Consumer, der die eingehenden Events aus Kafka parallel sowohl in ArcadeDB als auch in MicroRelate persistiert. Für die Anbindung von MicroRelate wurde ein zusätzlicher Adapter implementiert, der die internen Datenrepräsentationen des Loaders in das auf RDF und SHACL basierende Datenmodell von MicroRelate überführt. Die Konfiguration erfolgt deklarativ, wodurch es möglich ist, bestimmte Eventtypen gezielt nur einer der beiden Ziel-Datenbanken zuzuführen.

MicroRelate ermöglicht die strukturierte Persistierung und Abfrage von Knoten und Kanten, wobei zwischen Domänen-Objekten aus dem eigenen System und Foreign-Objekten aus Fremdsystemen unterschieden wird. Der Cartridge Loader nutzt einen dedizierten MicroRelate-Client, der über typisierte Repositories und einen zentralen EntityManager die Daten an die entsprechenden Endpunkte überträgt. EXP hat beide Systeme von Grund auf geplant, entwickelt und am Demonstrator getestet.

Für die Abfrage von MicroRelate wurde ein Cypher-Parser implementiert, der ein breites Subset der Cypher-Syntax unterstützt. Cypher ist eine deklarative Abfragesprache für Graph-Datenbanken, die ursprünglich für Neo4j entwickelt wurde und es erlaubt, komplexe Muster aus Knoten und Beziehungen intuitiv zu formulieren. Der Parser übersetzt Cypher-Abfragen in die native Abfragesprache von MicroRelate, wodurch Entwickler mit einer einheitlichen Abfragesprache arbeiten können, unabhängig davon welche Graph-Datenbank im Hintergrund angesprochen wird. Für die Integration mit LLM-basierten Suchfunktionen ist Cypher besonders vorteilhaft, da die strukturierte Syntax von LLMs zuverlässig aus natürlicher Sprache generiert werden kann.

## **Arbeitspaket 4: Kontextbasierte Enterprise Search**

### **AP 4.1: Aufbau des Suchkontexte**

Zu Beginn des Arbeitspakets 4 wurden zwei parallele Suchkonzepte erarbeitet, die komplementär eingesetzt werden sollten. Das erste Konzept setzte auf einen klassischen Suchalgorithmus mit einem konventionellen Suchfeld im User Interface, während das zweite auf Large Language Models (LLM) und ein Chatbot User Interface setzte. Diese duale Strategie wurde gewählt, um sowohl deterministische als auch semantische Suchanforderungen abzudecken und den Nutzern verschiedene Interaktionsmöglichkeiten zu bieten.

Nach einer ausführlichen Recherche der neuesten wissenschaftlichen Artikel und der Evaluation verschiedener Frameworks zur Integration von LLM-Technologie kam das Team zum Schluss, dass ein sehr modularer Ansatz verfolgt werden muss, um neue Entwicklungen im Bereich der LLMs so schnell wie möglich integrieren zu können. Die technische Evaluation und das Scouting des passenden Frameworks wurden von EXP übernommen. Die Entscheidung fiel dabei auf Langchain4j als zentrales Framework für die LLM-Integration. Diese Java-Bibliothek ermöglicht eine große Modularität bei gleichzeitig schneller und qualitativ hochwertiger Umsetzung. Langchain4j bietet eine einheitliche Abstraktionsschicht für verschiedene LLM-Anbieter und unterstützt Funktionen wie Prompt-Templates,

Memory-Management und Tool-Calling. Zudem integriert sich Langchain4j ideal in die bereits im Projekt etablierte Systemlandschaft und Arbeitsweise.

Für die klassische kontextbasierte Suche wurde ein Algorithmus entwickelt, der zwei ergänzende Ansätze miteinander kombiniert. Die Implementierung basiert auf der Graph-Struktur des Knowledge Graphs und ermöglicht die Bewertung und Priorisierung von Suchergebnissen anhand ihrer Verbindungsstruktur. Im Rahmen einer Masterarbeit wurde einer der Ansätze weiter untersucht, um eine Baseline für die Evaluation der LLM-basierten Suchergebnisse bereitzustellen.

Die Use Cases für beide Suchkonzepte wurden in einem gemeinsamen Workshop mit den Projektpartnern erarbeitet, der in Pfaffenhofen bei EXP durchgeführt wurde. Dabei wurden die technischen Rahmenbedingungen der Suchstrategien besprochen sowie die grobe Gestaltung des User Interfaces in Form von UI-Mockups definiert. Zusätzlich wurde Recherchearbeit zu den neuesten Visualisierungsbibliotheken für Graph-Strukturen durchgeführt, um eine optimale Darstellung der Suchergebnisse im Frontend zu ermöglichen. Diese frühe Einbindung aller Stakeholder stellte sicher, dass die entwickelten Suchfunktionen den tatsächlichen Anforderungen der Endnutzer entsprechen.

#### **AP 4.2: Konzeption der Suchfunktionen**

Das Herzstück der kontextbasierten Enterprise Search ist das Chat-Backend, das als vollwertiges LLM-Such-Interface entwickelt wurde. Die Architektur folgt einem modernen Ansatz, wie er von etablierten Chatbot-Anwendungen bekannt ist, wurde jedoch spezifisch auf die Anforderungen einer Enterprise-Umgebung zugeschnitten.

EXP implementierte ein Chatbot-Backend, das den gesamten Knowledge Graph des Projekts in natürlicher Sprache abfragen kann. Das System nutzt sogenanntes Graph Retrieval Augmented Generation (G-RAG), wobei vor allem eine Live-Abfrage der Datenbanken genutzt wird. Diese einzelnen Ergebnisse, die nun in natürlicher Sprache vorliegen, werden wiederum benutzt, um eine textuelle Antwort auf die Frage zu formulieren. Das Backend ist zudem in der Lage, den Kontext der vorherigen Nachrichten zu betrachten und Attribute aus dem Organisationsdiagramm wie den vollen Namen einer Person oder das Werk, in dem sie arbeitet, zu berücksichtigen.

Die Gesamtarchitektur des Chatbot-Systems umfasst mehrere zentrale Komponenten: Das MoInSe User Interface kommuniziert mit dem MoInSe AI Service, ebenfalls von EXP implementiert. Dieser greift einerseits auf den Vector Storage für die semantische Suche zu und andererseits über den Arcade CRT Ambassador Service auf die ArcadeDB. Der Ambassador Service fungiert dabei als Zwischenschicht, die die Kommunikation mit der Graph-Datenbank abstrahiert und zusätzliche Funktionen wie Caching und Metainformationsanreicherung bereitstellt. Die LLM-Komponente wurde zunächst mit dem von John Deere gehosteten OpenAI-Service verbunden.

Die Kommunikation zwischen Frontend und Backend erfolgt über WebSockets, was eine bidirektionale Echtzeitkommunikation ermöglicht. Dieser Ansatz wurde gewählt, da LLM-Antworten typischerweise in Form von Token-Streams generiert werden und Benutzer das Gefühl einer interaktiven Konversation erhalten sollen. Um eine sichere Kommunikation zwischen der Benutzeroberfläche und dem Backend sicherzustellen, wurde ein durch OIDC abgesicherter WebSocket implementiert. Dies stellt die Grundlage der angestrebten Fine-Grained Access Control (FGAC) dar, sodass Nutzer eindeutig identifiziert werden können und ihnen je nach Berechtigungen unterschiedliche Informationen zur Verfügung stehen.

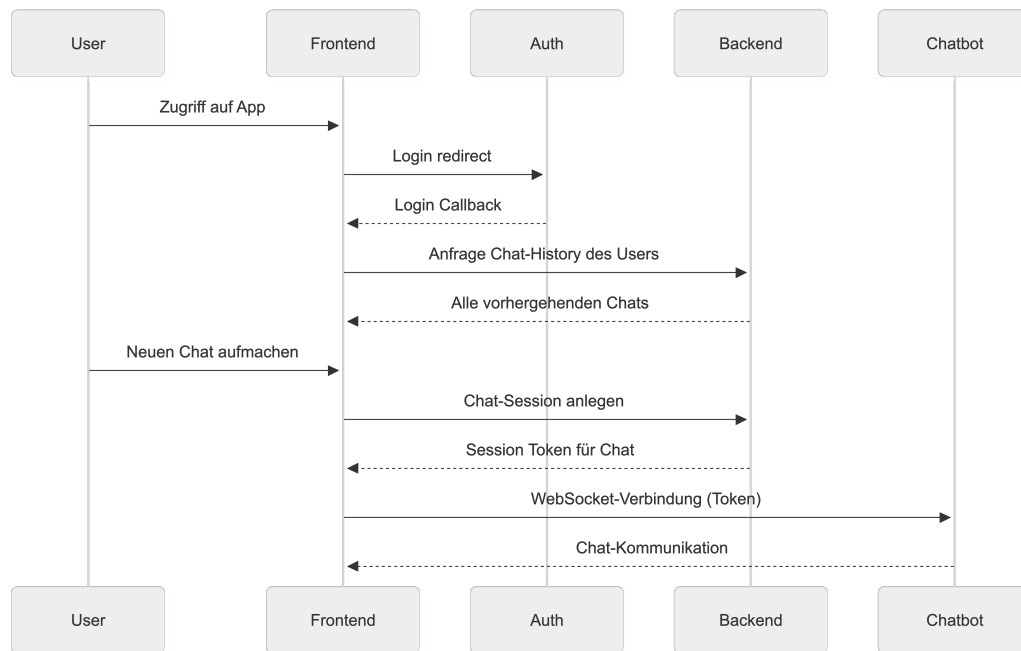


Abb. 4 Standardablauf für das Erstellen eines Chats via Chatbot

Das grafische User-Interface für den Chatbot wurde in enger Zusammenarbeit mit Fraunhofer IEM entwickelt. Es ermöglicht den Usern, ähnlich wie mit ChatGPT zu interagieren, und bietet somit eine intuitive Benutzererfahrung. Der erstellte UI-Prototyp wurde in der John Deere-Cloud deployt, um ihn dem Projektteam zum Testen und Erproben zur Verfügung zu stellen. Ein OIDC-abgesicherter Login in der Benutzeroberfläche wurde implementiert. Im Frontend wurde die Funktionalität der Suche um verschiedene Anzeigemöglichkeiten der Daten erweitert, darunter tabellarische Darstellungen und Graph-Visualisierungen. Dazu mussten im Chatbot-Backend von EXP entsprechende Anpassungen im Code wie auch des Promptings der LLM vorgenommen werden. Die Prompts wurden sorgfältig optimiert, um das LLM zur Generierung strukturierter Antworten anzuleiten, die sowohl in textueller Form als auch als Basis für Graph-Visualisierungen dienen können.

Eine zentrale Weiterentwicklung war die Implementierung der Chat-Persistenz. Diese Funktionalität ermöglicht es Benutzern, alte Konversationen wieder aufzunehmen und den darin enthaltenen Kontext wiederzuverwenden. Jeder Chat wird mit einer eindeutigen ID versehen und zusammen mit dem Benutzer-Identifizierer gespeichert. Wenn ein Benutzer einen Kontextwechsel vornehmen möchte, kann er einfach einen neuen Chat öffnen, ohne den Kontext des vorherigen Chats zu verlieren. Die Persistenzschicht wurde ebenfalls von EXP implementiert.

Die LLM-Anbindung durchlief mehrere Evolutionsstufen. Erste prototypische Anläufe wurden gegen die Cloud-API von OpenAI direkt implementiert, wobei Langchain4j als entkoppelnde Zwischenschicht diente. Anschließend erfolgte die Integration mit dem von John Deere selbst gehosteten OpenAI-Service (GPT-4o). Im Zuge der späteren Infrastrukturmigration wurde auf eine LLM-Instanz des Fraunhofer Instituts umgestellt, die eine vergleichbare Funktionalität mit höherer Integrationsfreiheit im Projektkontext bietet. Diese Umstellung bewies exemplarisch, dass der Softwarestack modular und gut wartbar ist. Es zeigte sich, dass der Demonstrator die Anforderungen an diverse LLM-Backends erfüllt.

Neben den Implementierungen der vordergründigen Zielsetzungen wurden diverse kleinere Dienste und Debug-Tools implementiert, welche unter anderem die Graph-Daten der ArcadeDB effektiv ausgeben konnten. Diese Hilfswerkzeuge erleichterten die Entwicklung und das Debugging des Gesamtsystems erheblich.

Im Zuge der Infrastrukturmigration auf AWS EKS wurde von EXP ein umfassendes Refactoring des Chat-Services durchgeführt. Statt der zuvor eingesetzten dedizierten Okta-Bibliothek kommt nun ein generischer OIDC-basierter Authentifizierungsflow auf Basis von Keycloak zum Einsatz. Diese Änderung war notwendig, da das System nach der Migration aus der John-Deere-Infrastruktur unabhängig von deren Identity-Provider betrieben werden musste. Keycloak übernimmt nun die

rollenbasierte Authentifizierung und Autorisierung der Benutzer und ist über Federation mit Microsoft Azure Active Directory als übergeordnetem Identity Provider verbunden. Dieser Aufbau gewährleistet ein hohes Maß an Sicherheit und ermöglicht gleichzeitig die Anlage von Testbenutzern, was für Entwicklung und Tests essenziell ist.

#### **AP 4.3: Implementierung des Such- und Indexalgorithmus**

Im abschließenden Arbeitspaket 4.3 wurde die Integration der Suchfunktionalität in den Federation Layer realisiert und ein zentrales Feature implementiert: der dynamische Switch zwischen den beiden Knowledge-Graph-Datenbanken ArcadeDB und MicroRelate. Diese Flexibilität war aus mehreren Gründen strategisch wichtig: Zum einen erlaubt sie die Validierung des Prototyps unabhängig von der zugrundeliegenden Persistenzschicht, zum anderen reduziert sie das Risiko eines Ausfalls bei der Weiterentwicklung einer einzelnen Graphdatenbank-Technologie.

Die Wahl von Cypher als Abfragesprache (s.o.) bietet einen entscheidenden Vorteil für die Integration mit LLM-Agenten: Die Syntax ist lesbar, strukturiert und konsistent, was sie ideal für die automatische Generierung von Abfragen macht. LLMs können Benutzeranfragen in natürlicher Sprache zuverlässig in Cypher-Queries übersetzen, ohne tiefgreifendes Vorwissen über die spezifische Datenbankstruktur zu benötigen. Beispielsweise kann eine Frage wie „Finde alle Bauteile, die mit einer Anforderung verbunden sind und von einem bestimmten Team freigegeben wurden“ direkt in eine entsprechende Cypher-Abfrage transformiert werden. Diese Eigenschaft macht Cypher zur optimalen Schnittstelle zwischen der KI-gesteuerten Suche und dem semantischen Wissen im Knowledge Graph.

Im Chat-Backend wurde von EXP die Logik für den Datenbankwechsel implementiert. Benutzer können über das Frontend die gewünschte Datenbank auswählen, woraufhin alle nachfolgenden Suchanfragen an die entsprechende Backend-Komponente geroutet werden. Die vom LLM generierten Cypher-Queries werden dabei entweder direkt an die ArcadeDB oder MicroRelate weitergeleitet, welche beide Cypher nativ unterstützt. Diese Abstraktion ermöglicht es, die Suchfunktionalität vollständig unabhängig von der darunterliegenden Graphdatenbank zu entwickeln und zu testen.

Zur Absicherung der Qualität des implementierten Such- und Indexalgorithmus wurde im Rahmen des AP4.3 ein eigenes Testing Framework mit dem Namen von EXP Rage4J entwickelt. Ziel war es, eine reproduzierbare und objektive Evaluierung von Such- und Rankingverfahren zu ermöglichen, da klassische Unit- und Integrationstests für Suchalgorithmen basierend auf natürlicher Sprache nur eine begrenzte Aussagekraft besitzen. Rage4J erlaubt die formale Definition von Testfällen bestehend auf Suchanfragen auf natürlicher Sprache, erwarteten Ergebnismengen und Relevanzgewichtungen je nach Methodik und eignet sich damit insbesondere für die Bewertung von Knowledge-Graph basierten Retrieval-Algorithmen.

Die Entwicklung von Rage4J erfolgte vollständig innerhalb des Projekts und wurde eng in die bestehende Java- und Quarkus-basierte Systemlandschaft integriert. Für ausgewählte, gemeinsam mit John Deere erarbeitete Use Cases wurden Referenzabfragen und Growth-Truth-Datensätze definiert, die als objektive Vergleichsbasis dienen. Auf dieser Grundlage berechnet Rage4J etablierte Metriken aus dem Information-Retrieval, wie Precision und Recall, sowie rankingbezogene Qualitätskennzahlen. Durch die Einbindung in die CI/CD-Pipeline konnten Änderungen am Such- und Indexalgorithmus sowie eine Änderung des LLMs kontinuierlich evaluiert und frühzeitig identifiziert werden. Rage4J stellte damit einen wesentlichen Baustein der Qualitätssicherung und Vergleichbarkeit der im Projekt entwickelten Suchansätze dar.

Die Entwicklung der Benutzeroberfläche für den MolnSe-Demonstrator erfolgte in enger Zusammenarbeit mit dem Fraunhofer-Institut. Gemeinsam wurde ein modernes, webbasiertes Frontend konzipiert und implementiert, das die Funktionalitäten des Knowledge Graphs für Endanwender zugänglich macht. Als technologische Basis wurde das JavaScript-Framework Vue.js gewählt, das sich durch seine Flexibilität, komponentenbasierte Architektur und gute Wartbarkeit auszeichnet.

Die Oberfläche ermöglicht die interaktive Visualisierung von Traceability-Beziehungen zwischen verschiedenen Engineering-Artefakten. Nutzer können durch verknüpfte Dokumente, Anforderungen und Designentscheidungen navigieren und so die Entstehungsgeschichte einzelner Systemkomponenten nachvollziehen. Darüber hinaus bietet das Frontend Suchfunktionen, mit denen gezielte Anfragen an den Knowledge Graph gestellt werden können. Die Ergebnisse werden übersichtlich aufbereitet und erlauben eine schnelle Orientierung in komplexen Datenbeständen.

Die partnerschaftliche Entwicklung erwies sich als besonders fruchtbar: Die EXP Software GmbH brachte ihre Expertise in der Entwicklung von Unternehmensanwendungen und der Integration von Backend-Systemen ein, während das Fraunhofer-Institut seine Forschungskompetenz im Bereich Wissensrepräsentation und Nutzerinteraktion beisteuerte. Durch regelmäßige Abstimmungen und iterative Entwicklungszyklen entstand eine Lösung für den Demonstrator, die sowohl den praktischen Anforderungen industrieller Anwender als auch wissenschaftlichen Ansprüchen gerecht wird.

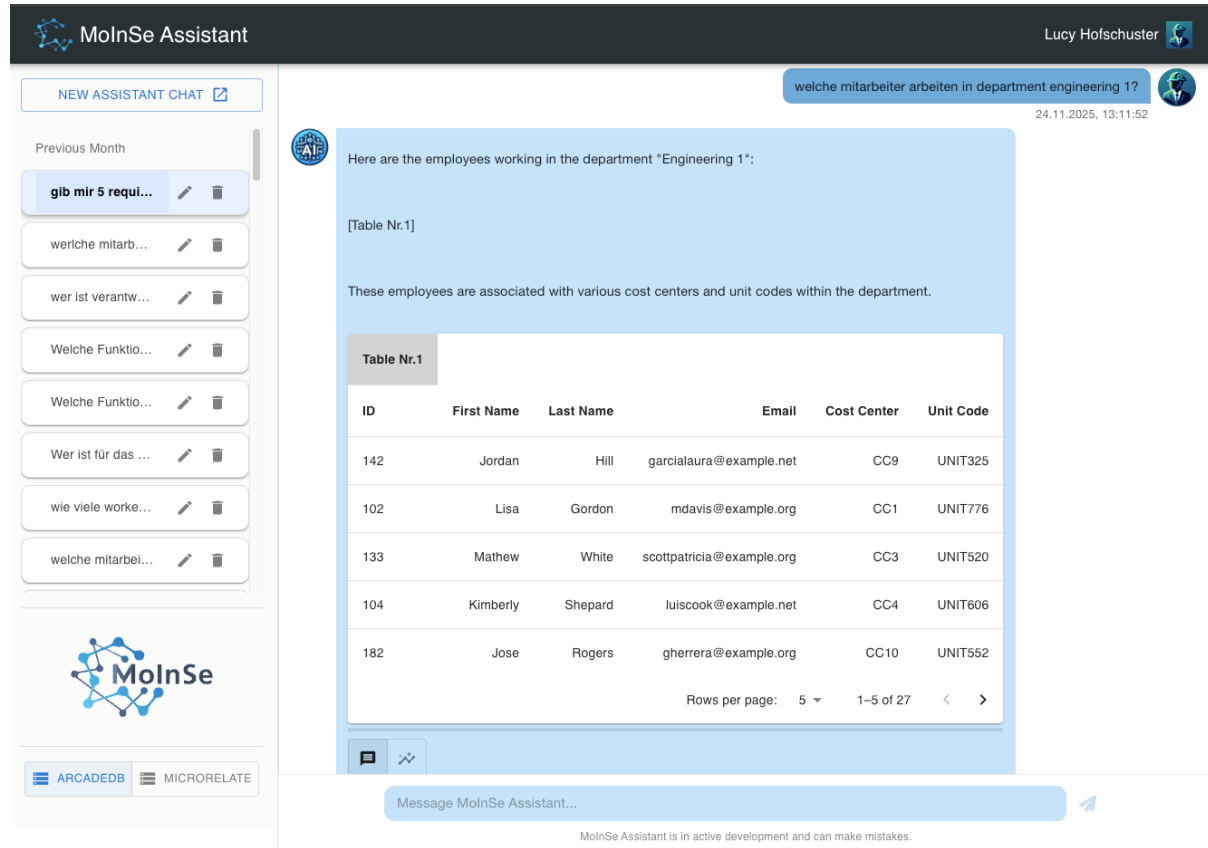


Abb. 5 Chat Interface welches der Enduser sieht mit Beispiel Nachricht

Die Arbeiten an AP 4 wurden erfolgreich abgeschlossen. Mit dem vollwertigen Chat-Backend, das eine moderne, chat-artige Benutzererfahrung bietet, und der Flexibilität, zwischen zwei verschiedenen Knowledge-Graph-Technologien wechseln zu können, wurden die Projektziele für die kontextbasierte Enterprise Search erreicht. Im Ganzen wurde die Chatbot-Funktionalität zu einem vollwertigen LLM-Such-Interface entwickelt, wie man es von gängigen Chatbot-Anwendungen kennt. Das resultierende Artefakt wurde in mehreren Workshops gemeinsam mit JD validiert, um die korrekte Funktionalität und Antwortgenauigkeit zu gewährleisten. Damit wurde eine intelligente und nachhaltige, aber auch korrekte Enterprise Search ermöglicht. Die im Projektverlauf gewonnenen Erkenntnisse zeigen, dass die Entwicklungen im Bereich der Generativen AI die Erreichung der ursprünglich geplanten Ziele sogar wahrscheinlicher gemacht haben als zum Zeitpunkt der Antragsstellung, da LLM-basierte Ansätze deutlich bessere Ergebnisse für semantische Suchanforderungen liefern als die ursprünglich geplanten rein algorithmischen Lösungen.

## Arbeitspaket 5: Modellbasierte Simulation des Entwicklungsprozesses

### AP 5.1: Integration der Process Engine und Definition des Musterprojekts

Ein zentrales Element des Arbeitspakets 5 war die Integration einer Process Engine in die MoInSe-Architektur. Das Ziel bestand darin, die in AP 1 modellierten BPMN-Prozesse nicht nur als statische Dokumentation zu betrachten, sondern sie tatsächlich zur Ausführung zu bringen und die dabei entstehenden Prozessdaten in den Knowledge Graph zu integrieren. Dieser Ansatz ermöglicht es, ein lebendiges Modell der Engineering-Prozesse zu schaffen, das kontinuierlich mit realen Ausführungsdaten angereichert wird.

EXP implementierte die CI/CD, die Konfiguration sowie Evaluation der ausgewählten Process Engine Kogito (s.o.).

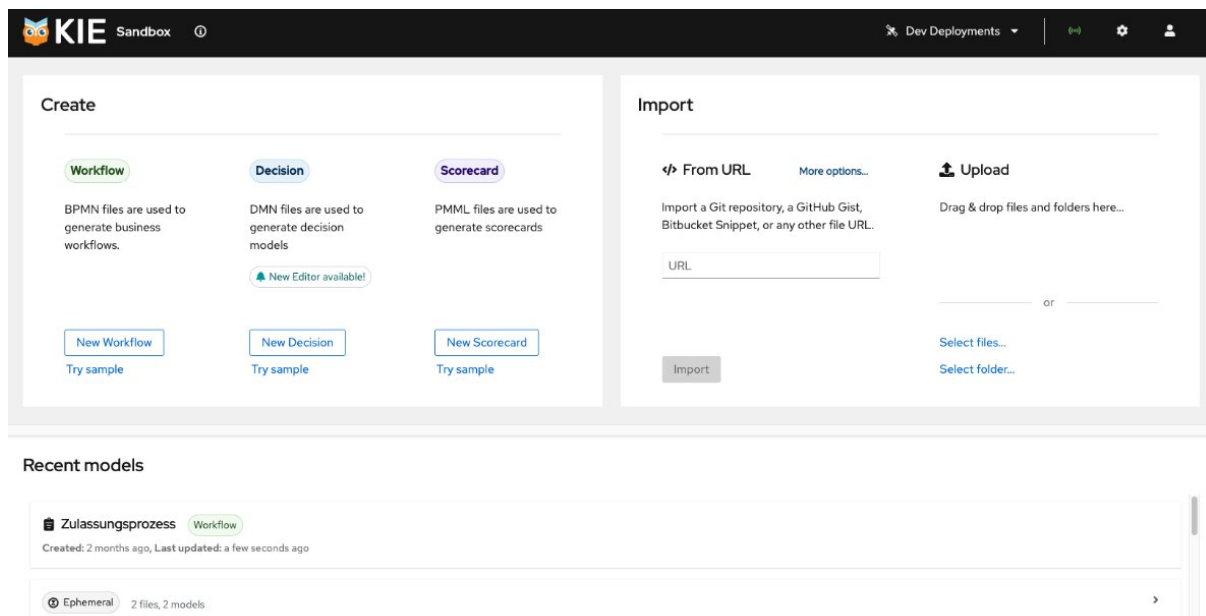


Abb. 6 KIE Sandbox zum Modellieren von Prozessen

Für das Musterprojekt wurden die Toy-Traktor-Daten verwendet, die im Rahmen von AP 2 bereits um mehrere hundert Testdatensätze erweitert worden waren. Die in AP 1 „Engineering Architecture Model“ aufgenommenen Prozesse wurden sorgfältig nach empirischen Kriterien untersucht, um eine Auswahl zu treffen, die programmatisch in der Process Engine ausgeführt werden kann. Aus dieser Analyse entstand die Entscheidung, zunächst einen gewissen Prozess als ausführbaren Prozess zu implementieren.

Dieser BPMN-Prozess modelliert einen typischen Change-Management-Workflow. Er beginnt mit einem eingehenden Engineering Change Request, der eine Konfigurationsänderung erfordert. EXP übernahm hier die Implementierung des ausführbaren Prozesses und die Überführung sowie das Deployment in die Process Engine.

Die Ausführung dieses Prozesses in Kogito erzeugt echte Prozess-Metadaten für den Knowledge Graph. Die Ereignisse werden als strukturierte Daten erfasst und können mit den zugehörigen Engineering-Artefakten im Knowledge Graph verknüpft werden. Die Management-Console der Engine visualisiert dabei den aktuellen Prozesszustand, zeigt die Timeline der durchlaufenen Aktivitäten und ermöglicht das manuelle Triggern von Knoten für Testzwecke. Die Prozessvariablen, wie etwa die ID, der Name, der Status und das zugehörige Datum, werden als JSON-Struktur gespeichert und stehen für die weitere Verarbeitung zur Verfügung.

Dieser „Process First“-Ansatz bietet einen entscheidenden Vorteil gegenüber herkömmlichen Analysemethoden: Er erübrigt es komplett, A-Posteriori-Verfahren wie Process Mining einsetzen zu müssen. Während Process-Mining-Tools darauf angewiesen sind, Prozesse aus bestehenden Log-Daten zu rekonstruieren, werden bei unserem Ansatz die Prozesse von vornherein strukturiert ausgeführt und dokumentiert. Dies führt zu einer höheren Datenqualität und ermöglicht eine nahtlose Integration der Prozessdaten in den Knowledge Graph. Die Möglichkeit, das Musterprojekt zu erweitern, wurde bewusst offengehalten, um im Demonstrator diese exemplarische Quelle an Daten anzubinden und bei Bedarf weitere Prozesse hinzuzufügen.

#### AP 5.2: Definition von Planungsziel und Simulationsparametern

Im Rahmen des Arbeitspakets 5.2 wurde ein innovativer Ansatz zur Simulation von Engineering-Prozessen entwickelt, der auf der Kombination von BPMN-Prozessen mit Large Language Models basiert. In einer Bachelorarbeit an der Technischen Hochschule Ingolstadt wurde untersucht, wie gut Prozesse mit LLMs simuliert werden können. Das zentrale Ziel bestand darin, möglichst viele realistische Daten für den Knowledge Graph zu erzeugen und aus der Simulation einen Forecast über

wichtige Eigenschaften der tatsächlichen Prozessinstanzen ableiten zu können. EXP übernahm hier die Planung, die Implementierung und Optimierung in enger Zusammenarbeit mit dem Absolventen.

Der entwickelte Ansatz basiert auf einer sogenannten Agenten-Architektur. Die einzelnen Akteure in den BPMN-Prozessen werden dabei durch LLM-Agenten abgebildet, die ihre eigene „Persönlichkeit“ besitzen und so eine glaubwürdige Simulation der Abläufe in einem Unternehmen garantieren. Diese Agenten basieren auf einem SaaS-LLM und wurden über die Java- in die Kogito-Prozessumgebung eingebunden. Über strukturierte JSON-Daten werden Ziele, Kontexte und Rolleneigenschaften an die Agenten übergeben, sodass sie kontextbezogen und rollenspezifisch agieren können.

Im Rahmen der Bachelorarbeit wurden drei exemplarische BPMN-Prozesse unterschiedlicher Komplexität implementiert, die analog zu den Prozessen aus den früheren Arbeitspaketen gestaltet wurden. Diese reichen von einfachen Planungsaufgaben über die Klassifikation von Tickets bis hin zu iterativen Design- und Bewertungszyklen mit Feedback-Loops. Das System wurde hinsichtlich verschiedener Qualitätsmetriken evaluiert, darunter Ausführungszeit, Kosten, Entscheidungskonsistenz und Iterationsverhalten. Besonders im iterativen Szenario zeigte sich, dass das Verhalten der Agenten stark von ihrer charakterlichen Beschreibung beeinflusst werden kann. Diese Erkenntnis ermöglicht eine gezielte Parametrisierung in zukünftigen Anwendungen, um unterschiedliche Verhaltensweisen und Entscheidungsmuster zu simulieren.

Die durch die LLM-Agenten erzeugten Prozesse generieren realistische Datenartefakte, die zurück in das IT-System gespielt werden können. Dadurch lässt sich das Musterprojekt um die zeitliche Achse erweitern und ein Forecast erstellen. Ein besonderer Vorteil des Ansatzes ist, dass auch die zeitliche Achse simuliert wird: In einem kleinen realen Zeitraum können Daten erzeugt werden, die einem großen simulierten Zeitraum entsprechen. Dies ermöglicht es, Szenarien zu testen, die sich in der Realität über Monate oder Jahre erstrecken würden.

### **AP 5.3: Aufbau des Prototypen**

Die Ergebnisse der Bachelorarbeit zur LLM-basierten Prozesssimulation liefern wertvolle Erkenntnisse für das MoInSe-Projekt. Der entwickelte Ansatz zeigt, wie durch die Kombination von BPMN-Prozessen und LLM-Agenten synthetische, aber semantisch reichhaltige Prozessdaten erzeugt und analysiert werden können. Diese umfassen Prozessinstanzen, Entscheidungspunkte und simulierte Nutzerinteraktionen, die ein realistisches Abbild von Produktentwicklungsprozessen mit menschlichem Entscheidungsanteil ermöglichen – ohne auf reale, sensible Nutzerdaten angewiesen zu sein. EXP übernahm hier die Implementierung, die Persistenz und die Auswertung der Ergebnisse.

Die Kombination von BPMN und LLMs erlaubt wiederholbare und kostengünstige Prozesssimulationen. Besonders für frühe Entwurfs- und Evaluationsphasen bietet dieser Ansatz Vorteile, da keine reale Nutzerinteraktion erforderlich ist. Die Struktur der generierten Daten eignet sich grundsätzlich für die Integration in einen Knowledge Graph, der sowohl regelbasierte als auch KI-basierte Traceability-Analysen unterstützen kann.

Die Process Engine dient dabei als zentrale Orchestrierungsschicht. Sie koordiniert die Ausführung der BPMN-Prozesse, verwaltet die Prozesszustände und stellt die Schnittstellen für die LLM-Agenten bereit. Über Kafka-Events können Prozessinstanzdaten an nachgelagerte Systeme wie den Cartridge Loader übermittelt werden, der sie in den Knowledge Graph überführt.

Die Implementierung der Simulation blieb im Rahmen des Projekts prototypisch und wurde nicht vollständig in den Demonstrator integriert. Dennoch wurden konkrete Schritte unternommen, um die Process Engine tiefer in das MoInSe-Ökosystem einzubinden. Hierzu zählt insbesondere das Deployment der KIE Sandbox, mit der BPMN-Prozesse direkt im Browser modelliert und deployt werden können. Diese Infrastruktur schafft die Grundlage für eine weiterführende Integration in zukünftigen Projektphasen.

Der gewählte Ansatz demonstriert, dass die Integration von Process Engines mit modernen KI-Technologien ein vielversprechender Weg ist, um die Komplexität von Engineering-Prozessen beherrschbar zu machen. Die prototypischen Ergebnisse zeigen das Potenzial für die Generierung realistischer Testdaten und die Optimierung von Entwicklungsabläufen auf, indem durch Forecast-Simulationen eine Aussage über die tatsächliche Ausführung gemacht werden kann.

### **Arbeitspaket 6: Projektmanagement / Transfer / Datenmanagement**

Im Rahmen des Arbeitspakets 6 beteiligte sich EXP an den für das Projektmanagement relevanten Artefakten des Scrum-Prozesses, insbesondere an den wöchentlichen Refinement- und Sprint-Review-

Meetings sowie den separat stattfindenden Retrospektiven. Regelmäßig wurden Anforderungen von John Deere erhoben, analysiert und zur Implementierung vorbereitet. Zusätzlich fanden mehrmals zweitägige Implementierungswrkshops statt, an denen alle Projektpartner gemeinsam fokussiert am Projekt gearbeitet haben.

Ein wesentlicher Schwerpunkt lag auf der Anonymisierung und Obfuskation von Daten. Da das Softwaresystem nach der Infrastrukturmigration außerhalb der John-Deere-Systemlandschaft betrieben wird, wurde ein formaler Prozess zur Datenfreigabe initiiert. Dabei wurden alle aus Quell- und Mocksystemen eingebundenen Datenquellen klassifiziert, dokumentiert und technisch aufbereitet. Die zugehörigen Datenstrukturen konnten in weiten Teilen als unkritisch eingestuft und zur Nutzung freigegeben werden. Die enthaltenen Daten selbst mussten jedoch gemäß Datenschutzerfordernngen anonymisiert und inhaltlich verfälscht werden. Dies wurde durch einen speziell entwickelten Java-Service sichergestellt, der strukturierte Anonymisierung und gezielte Verfälschung technischer Details automatisiert durchführt. Auf diese Weise konnten realitätsnahe, aber DSGVO-konforme Test- und Demonstrationsdaten für die weitere Projektarbeit bereitgestellt werden.

Im Bereich des Wissenstransfers wurde im Rahmen des Projekts an mehreren wissenschaftlichen Arbeiten mitgewirkt. Eine Bachelorarbeit an der Technischen Hochschule Ingolstadt behandelte die „Entwicklung und Evaluation eines Vorgehensmodells für die Entwicklung von Geschäftsprozessen im Engineering“ und lieferte wertvolle Grundlagen für die Prozessmodellierung in AP 1. Eine weitere Bachelorarbeit untersuchte die Simulation von BPMN-Prozessen mithilfe von LLM-Agenten, deren Ergebnisse direkt in AP 5 eingeflossen sind. Im Rahmen einer Masterarbeit wurde ein umfassendes Konzept zur Historisierung von Wissensgraphen entwickelt und evaluiert, das die Grundlage für den Temporal Property Graph in AP 2 bildete. Eine weitere Masterarbeit widmete sich dem gewichteten Page-Rank-Algorithmus für die kontextbasierte Suche in AP 4. Darüber hinaus wurde mit der Arbeit an einem wissenschaftlichen Paper begonnen, das die AI-gestützte Generierung von Tracelinks im Rahmen eines Engineering-Projekts untersucht.

Name	Titel
Philipp Pfaller	“Entwicklung und Evaluation eines Vorgehensmodells für die Entwicklung von Geschäftsprozessen im Engineering”
Sascha Shakirin	“Evaluation verschiedener Historisierungsmethoden für typisierte Wissensgraphen”
Viktor Huska	“Conceptualization and Implementation of AI-Driven Agents in BPMN Processes”
Gabriel Popa	“Context-Sensitive Result Weighting and Result Ordering in Knowledge Graph-Based Search Queries”

Im Mai 2025 hat die EXP Software GmbH die Rechte und Pflichten der exentra GmbH als Teilprojekt-Inhaber übernommen. Die EXP Software GmbH ist ein 100-prozentiges Tochterunternehmen der exentra GmbH. Die bislang am Forschungsprojekt beteiligten Entwickler sind nun im Tochterunternehmen beschäftigt und stehen dem MolnSe-Projekt weiterhin mit ihrem Know-how zur Verfügung. Da die gleichen Personen an Bord blieben, hat sich der Übergang für das Projektgeschehen als nicht spürbar vollzogen.

## 2. Wichtigste Positionen des zahlenmäßigen Nachweises

Der größte Kostenblock innerhalb des Projektrahmens entfiel erwartungsgemäß auf die Personalkosten. Hinzu kommen Reisekosten, die einen Anteil von rund 1 % ausmachen, sowie sonstige

Kosten, die sich vorrangig aus Lizenzgebühren, Arbeitsgeräten und Cloud-Diensten zusammensetzen. Letztere haben im Verlauf des zurückliegenden Projektjahres infolge der Migration der IT-Infrastruktur auf ein von EXP verwaltetes Kubernetes-Cluster einen Anstieg von bis zu 10 % verzeichnet. Die Mittelverwendung entspricht damit dem für ein Projekt dieser Art typischen Muster. Der Anstieg der sonstigen Kosten ist auf strukturelle Veränderungen innerhalb eines Konsortialpartners zurückzuführen. Darüber hinaus ist anzumerken, dass der verstärkte Einsatz KI-gestützter Entwicklungswerkzeuge im letzten Projektjahr zu einer spürbaren Reduzierung des Entwicklungsaufwands beigetragen hat.

### **3. Notwendigkeiten und Angemessenheiten der geleisteten Arbeiten**

Die im Projekt erarbeiteten Artefakte umfassen viele Bereiche, was sich durch die Tiefe und den Umfang des Projekts begründet. Das Ziel, welches in der Vorhabensbeschreibung gesetzt wurde, war es, die Prozesse, Systeme und weitere Aspekte der Fahrzeugentwicklung eingehend zu analysieren und eine Plattform zu entwickeln, die alle relevanten Informationen zusammenträgt und abfragbar macht. John Deere weist genau diese Komplexität durch seinen globalen Charakter und die hohe Produktkomplexität auf, welche wiederholt zu Ineffizienzen und Problemen führen kann. Dafür haben wir uns intensiv mit verschiedenen Themen der Fahrzeugentwicklung in enger Zusammenarbeit mit den jeweiligen Domänenexperten auseinandergesetzt, um am Projektende einen aussagekräftigen Demonstrator vorweisen zu können.

Die im Projekt erarbeiteten Softwareartefakte bzw. der Demonstrator als Ganzes versucht genau diese Komplexität durch eine neuartige Kombination sowohl neuer als auch bewährter grundlegender Technologien zu vereinfachen. Hierbei wird beispielsweise eine Graph-Datenbank mit Fremdverweisen als Federation Layer (AP 2) eingesetzt, um den Zusammenhang verschiedener verwandter Datenstämme aus Fremdsystemen zu korrelieren. Dies wurde mithilfe des in Teil II 1) angesprochenen ereignisbasierten Imports umgesetzt. Die darauffolgende Verknüpfung der Daten (AP 3) durch das Trace-Linking sowie die Durchsuchbarkeit mittels des KI-gestützten Chatbots (AP 4) machen den MoInSe-Demonstrator zu einem wertvollen Werkzeug in der Fahrzeugentwicklung.

Jedes einzelne Artefakt des Demonstrators erfüllt eine wichtige Aufgabe bzw. Anforderung, die zum Gesamtergebnis beiträgt. Auch die Arbeiten im Rahmen von AP 5 sind hier zu erwähnen, welche zwar nicht in eine Forecast Engine mündeten, den Demonstrator jedoch mit den Toytraktor-Daten versorgten. Weitere Arbeiten entstanden im Zuge der Migration der Infrastruktur, die nicht vermeidbar waren. Hinzu kommen die in modernen Softwareumgebungen gängigen Hintergrundsysteme wie beispielsweise Monitoring-Dashboards.

Die aufgewendeten Ressourcen stehen in einem angemessenen Verhältnis zum resultierenden Demonstrator. Die Arbeiten, die stets darauf ausgerichtet waren, die angestrebten Forschungsergebnisse zu erzielen, wurden effizient, zielorientiert und entsprechend dem genehmigten Arbeits- und Zeitplan durchgeführt.