Wilhelm Büchner Hochschule
Hilpertstr. 31
D-64295 Darmstadt

Data Science Projekt

Fachbereich Informatik

Process Mining im Umfeld Abas ERP

Betreuer:	Hr. Coskun Akinalp
Autor:	Manuel Ott
Matrikelnummer:	907778
Anschrift:	Freiheitstraße 46 42277 Wuppertal
Abgabetermin:	29. Juni 2026

Zusammenfassung

Die vorliegende Projektarbeit untersucht die Umsetzung einer Process Mining Methodik im Umfeld des Enterprise Resource Planning (ERP)-Systems Abas ERP. Ziel ist es, insbesondere mittelständischen Unternehmen datengetriebene Prozessanalyse zugänglich zu machen, ohne auf kommerzielle und oftmals unrentable Tools zurückgreifen zu müssen. Aufgrund der proprietären Datenbankarchitektur und dynamischen Datenbankschemata gestaltet sich die Datenextraktion und -transformation in diesem Umfeld als besonders komplex.

Durch die zunehmende Entwicklung künstlicher Intelligenz entstehen Synergien in ebendieser Methodik, die Process Mining Projekte vereinfachen und damit wirtschaftlicher und zugänglicher für mittelständische Unternehmen machen. Durch einen kooperativen Künstliche Intelligenz (KI)-Ansatz werden deterministische und replizierbare Phasen mit gezielter Einbindung eines KI-Agenten für das Schema-Mapping kombiniert. Der Mensch hat dabei die Entscheidungsgewalt und muss die vom Agenten generierten Vorschläge explizit übernehmen.

Die technische Umsetzung basiert auf einer Glass Box Architektur und ist unterteilt in vier Phasen. Dabei stellt jede Phase ein Artefakt bereit, welches als Grundlage für die Folgephase dient. Jede Phase ist damit maximal transparent und reproduzierbar. Das Artefakt der letzten Phase ist ein valides eXtensible Event Stream (XES)-Log, welches folgend für tiefreichende Prozessanalysen durch dedizierte Tools genutzt werden kann.

Abstract

This project paper examines the implementation of a process mining methodology within the context of the ERP system Abas ERP. The goal is to make data-driven process analysis accessible particularly for small and medium-sized enterprises without them relying on commercial and often cost-prohibitive tools. Data extraction and transformation are especially complex due to the proprietary database architecture and dynamic database schemas of Abas ERP.

Advances in artificial intelligence are creating synergies within this methodology, simplifying process mining projects and making them more cost-effective and therefore accessible for small and medium-sized enterprises. A collaborative Artificial Intelligence (AI) approach combines deterministic and replicable phases with the targeted integration of an AI agent for schema mapping. Human oversight is still mandatory, requiring explicit approval of the suggestions generated by the agent.

The technical implementation is based on a glass box architecture and is divided into four phases. Each phase produces an artifact that serves as the foundation for the subsequent phase, ensuring maximum transparency and reproducibility. The artifact resulting from the final phase is a valid XES log, which can subsequently be used for in-depth process analysis via dedicated tools.

Inhaltsverzeichnis

1. Einleitung

Process Mining ist eine etablierte Methodik, um Geschäftsprozesse datengetrieben zu modellieren. Mit einem digitalen Prozesszwilling können folgende Überprüfungen hinsichtlich Konformität (Ist-Zustand gegen Soll-Zustand), beispielsweise durch zuvor definierte Business Process Management (BPM)-Modelle stattfinden. Ferner kann aus dem rekonstruierten Modell eine Vielzahl an Key Performance Indicators (KPIs) ermittelt sowie Engpässe und damit Optimierungspotenziale aufgedeckt werden. Klassische KPIs sind beispielsweise die Liegezeit[1] oder die Bearbeitungszeit[2] (Hornix, 2007, S. 29–30).

Im Umfeld des ERP-Systems „Abas ERP“ bietet die Methodik des Process Mining enormen Mehrwert. Das ERP-System bietet als zentrale Plattform für Unternehmensprozesse ein hervorragendes Einsatzgebiet für Process Mining.

Aufgrund verschiedener technischer Einschränkungen gestaltet sich die Anwendung des Process Mining im Umfeld Abas ERP jedoch recht kompliziert. Mit einer eigenen Datenbankarchitektur und zugehörigem, proprietären Protokoll, sowie teils hochindividuellen Variablen und Datenbankstrukturen bedarf es hoher Flexibilität beim Extrahieren und Auswerten der Daten.

[1] vergangene Zeit zwischen dem Abschluss einer Aktivität und dem Start der Folgeaktivität
[2] benötigte Zeit für die Durchführung einer Aktivität

Abas ERP ist ein ERP-System, entwickelt von der Forterro Deutschland Abas GmbH in Karlsruhe und spezialisiert sich auf fertigende Unternehmen im Mittelstand (Forterro Deutschland Abas GmbH, 2026). Neben den technischen Einschränkungen ergeben sich damit auch unternehmerische Einschränkungen. Da Process Mining bzw. Data Mining im Allgemeinen kein Selbstzweck ist, muss auch der Aufwand für solche Projekte berücksichtigt werden.

Die vorliegende Projektarbeit hat zum Ziel, verschiedene Ansätze des Process Mining im Umfeld Abas ERP zu validieren und maßgeblich zu definieren, wie Process Mining Projekte künftig umgesetzt werden. Insbesondere durch die gegenwärtige Entwicklung von KI sind in diesem Gebiet viele Potenziale aufgekommen, auf die in dieser Projektarbeit gezielt eingegangen wird.

1.1. Projektbeschreibung

Das Projekt gliedert sich in zwei wesentliche Bestandteile: Einen *Forschungsteil*, der drei alternative Ansätze für Process Mining durchleuchtet und einen *Implementierungsteil*, der wiederum das Ergebnis des Forschungsteils konkret in einem Projekt umsetzt. Eine zentrale Forschungsfrage dreht sich um den Einsatz von KI für Process Mining. Der Forschungsteil gliedert sich daher in die folgenden drei Ansätze auf:

1. **Vollständiger und autonomer Einsatz von KI**: Die gesamte ETL-Pipeline (Extract, Transform, Load) und jegliche Auswertungen finden autonom über Prompts und ohne weiteres involvieren eines Menschen statt.

2. **Kooperativer Einsatz von KI**: KI dient als Unterstützung für das gesamte

Process Mining Projekt.

3. **Verzicht auf den Einsatz von KI**: Klassisches Process Mining ohne Unterstützung durch KI.

Diese Ansätze werden im weiteren Verlauf dieser Projektdokumentation ausführlich beschrieben.

1.2. Umfeld

1.2.1. Organisatorisches Umfeld

Dieses Projekt wurde im Rahmen des Studiengangs „Angewandte Informatik“ an der Wilhelm Büchner Hochschule in Darmstadt in der Vertiefungsrichtung „Data Science“ durchgeführt.

Das Projekt wurde in Kooperation mit der MAIT Gruppe (MAIT Gruppe, 2026) umgesetzt. Die MAIT-Gruppe ist führender Digitalisierungspartner für den europäischen Mittelstand und begleitet die Umsetzung und Optimierung digitaler Geschäftsprozesse. Mit führenden Softwarelösungen in den Bereichen Product Lifecycle Management (PLM), ERP sowie individuellen Informationstechnologie (IT)- und Cloudservices verfolgt sie einen ganzheitlichen Digitalisierungsansatz: „Alles aus einer Hand“ (MAIT Gruppe, 2026).

1.2.2. Technisches Umfeld

Abas ERP ist im internen Netzwerk des Kunden („On-Premise“) gehostet und baut auf dem Linux Derivat Red Hat Enterprise Linux (RHEL) (Red Hat, Inc., 2026) auf. Die interne Netzwerkkommunikation findet via ERP Datenübertragungsprotokoll (EDP) auf Port 6550 (Transmission Control Protocol (TCP)) statt. Dabei findet die Installation des gesamten ERP-Systems auf einem einzelnen Server statt.

Je nach Kundeninstallation variiert der Systemumfang des Servers stark. Das betrifft sowohl die Hardwarespezifikationen, wie die Größe des Random Access Memory (RAM) oder die Anzahl an Central Processing Unit (CPU)-Kernen, als auch die Datenbankgröße in Form der rohen Datenmenge.

Für die Kommunikation mit Abas ERP stellt die Forterro Deutschland Abas GmbH neben dem reinen EDP noch zusätzliche Software zur Verfügung. Das umfasst:

1. Das Shellskript **edpexport.sh**
2. Das **Entwicklungsframework** Abas Java Objects (AJO)
3. Die Abas **Middleware** (Representational State Transfer (REST)-Application Programming Interface (API))

Auf diese Technologien wird im Kapitel „Ideenfindung“ weiter eingegangen.

Die ERP-Software lässt sich selbst wiederum in zwei Bereiche teilen. Es gibt den sogenannten „Kern“ und die „Flexible Oberfläche“. Abas ERP ist eine Software die von Grund auf Individualisierungsmöglichkeiten anbietet und sogar darauf ausgelegt ist. Während der Kern statisch (nicht anpassbar) ist, bietet die Flexi-

ble Oberfläche maximale Anpassungsmöglichkeiten. So ist es möglich, tief in die Software einzugreifen und eigenentwickelte Programme (Flexible Oberflächenprogramme (FOPs)) eventgesteuert zu hinterlegen. Ferner kann jede Datenbanktabelle nach belieben erweitert werden oder gänzlich neue Datenbanktabellen inklusive Fremdverweisen erstellt werden und damit selbst spezifischste Prozesse umgesetzt werden.

1.3. Zielsetzung

Der Ansatz dieser Arbeit im Allgemeinen ist es, auch mittelständischen Unternehmen die Zugänglichkeit zu Process Mining zu ermöglichen ohne zusätzliche (oftmals nicht rentable) kommerzielle Tools lizenzieren zu müssen. Damit ist eines der Ziele die Wirtschaftlichkeit in Form von hoher Effektivität und Effizienz der künftig umzusetzenden Process Mining Projekte im Allgemeinen.

Da die vorliegende Projektarbeit mehrere Ansätze für künftige Process Mining Projekte validiert, unterteilt sich die Zielsetzung selbst in zwei wesentliche Bestandteile: (1) Die Anforderungen an künftig umzusetzende Process Mining Projekte und (2) die daraus abgeleiteten Ziele des in der vorliegenden Projektarbeit beschriebenen Forschungsanteils.

1.3.1. Allgemeine Ziele von Process Mining Projekten im Umfeld Abas ERP

Vor Beginn dieser Projektarbeit wurden Ziele und Anforderungen der Process Mining Projekte definiert. Diese umfassen:

End-to-End Transparenz über alle Prozesse hinweg schaffen Es soll der gesamte Prozessablauf der Geschäftsvorgänge nachvollziehbar gemacht werden. Das bedeutet, ein Prozess soll von Anfang bis Ende sowohl über alle Abteilungen (organisatorisch) als auch über alle Systeme (technisch) hinweg transparent gemacht werden. Dadurch sollen Datensilos aufgelöst und „blinde Flecken“ aufgedeckt werden.

Prozessabweichungen erkennen und Optimierungspotenziale ableiten Durch ein Soll-/Ist Abgleich werden Abweichungen in den gelebten Prozessen aufgedeckt. Mithilfe weiterer Metriken, wie Liegezeiten oder Kosten, werden gewichtete Optimierungspotenziale erschlossen. Darunter fallen beispielsweise das Aufdecken von Engpässen oder Häufungen manueller oder abweichender Prozessschritte in bestimmten Situationen.

Prozesse datenbasiert steuern durch Reporting und Analytics Durch den datenbasierten Ansatz sollen Entscheidungen und Prozesse faktenbasiert (objektiv) statt subjektiv erfolgen. Mithilfe von Analytics und Reporting Tools sollen die operativen Daten dynamisch und interaktiv ausgewertet werden, um diese Entscheidungen zu stützen.

Aufbau eines digitalen Prozesszwillings für unternehmensweite Transparenz Durch die kontinuierliche Integration aktueller, operativer Daten soll ein dynami-

sches, virtuelles Abbild der realen Prozesse geschaffen werden. Dazu sollen nicht nur die Prozessschritte, sondern auch die unterliegenden Entscheidungen und Zusammenhänge transparent gemacht werden. Dies dient als Fundament für weitere Analysemöglichkeiten, wie etwa Vorhersagen oder dem Aufdecken von Anomalien durch Modelle künstlicher Intelligenz.

1.3.2. Forschungsziele und Anforderungen an die vorliegende Projektarbeit

Aus den zuvor genannten Zielen an Process Mining Projekten im Allgemeinen lassen sich für die vorliegende Projektarbeit damit folgende zentrale Anforderungen und Ziele ableiten:

1. **Vereinfachung/Automatisierung der Extract, Transform, Load (ETL)-Pipeline** Eine ETL-Pipeline teilt sich auf in die drei Phasen (1) Extraktion (**E**xtract), in der der Datenfluss durch direkten Zugriff auf die Datenquellen der operativen Systeme (im vorliegenden Fall vorerst nur das ERP-System) stattfindet; (2) Transformation (**T**ransform), in der die extrahierten Daten dann harmonisiert werden und in ein einheitliches, allgemein umgängliches Datenschema überführt werden, damit sie dann (3) geladen (**L**oad) und somit für Prozessanalysen genutzt werden können.

   Dabei sind die extrahierten Daten oft sehr heterogen. Sie weisen oft verschiedenste Formate (wie unterschiedliche Datumsformate) und interne Datentypen auf. In diesem Projekt sollen die heterogenen Daten aus Abas ERP direkt während der ETL-Pipeline in ein einheitliches Format überführt werden.

2. **Berücksichtigung hochindividueller Datenstrukturen und Prozesse** Eine besondere Herausforderung stellen die vielschichtigen Individualisierungsmöglichkeiten dar. Es muss daher eine Logik geben, um individuelle Prozesse und die damit zusammenhängenden Datenstrukturen zu identifizieren und damit bestenfalls sogar Abhängigkeiten zwischen den Daten (in Form von Verweisen) zu erkennen.

3. **Erzeugung von XES konformen Event Logs** Für eine tiefere Analyse sollen Prozessdaten in das standardisierte Format XES überführt werden. Dafür müssen die Daten für jegliche (auch für individuelle) Prozesse identifiziert und in die entsprechenden Zielfelder transformiert werden.

2. Ideenfindung

Die Ideenfindung stellte den zeitintensivsten Anteil des Projekts dar. Da es firmenintern erst wenig Erfahrung oder Kompetenz im Bereich Process Mining gibt, musste vorerst der grundlegende Rahmen definiert werden.

Dieser umfasst die Aspekte Einsatz von KI in drei Ausprägungen, die Technologie für die Datenextraktion und alle weiteren Technologieentscheidungen, wie die Wahl der Programmiersprache oder der richtigen Frameworks.

2.1. Einsatz von KI

Die zentrale Fragestellung beim Einsatz von KI war, an welchen Stellen der Pipeline KI sinnvoll eingesetzt werden kann und an welchen Stellen deterministische oder statistische Verfahren vorzuziehen sind. KI ist insbesondere in dynamischem Umfeld nützlich und kann sich auf individuelle Prozesse und Strukturen adaptieren. Allerdings müssen Ergebnisse, die auf diese Art entstehen hinterfragt und verifiziert werden. Insbesondere Halluzinationen stellen eine große Herausforderung dar (Siebert, 2024). Die nicht-deterministische Natur künstlicher Intelligenz schränkt zudem die Reproduzierbarkeit der Ergebnisse ein und erschwert damit

nicht nur die Nachvollziehbarkeit der Process Mining Pipeline, sondern auch der Forschungsergebnisse dieser Projektarbeit.

2.1.1. Ansatz 1: Vollständiger Einsatz von KI

Der erste Ansatz sah vor, die gesamte Process Mining Pipeline und alle darauf aufsetzenden Analysen vollständig über einen oder mehrere Agenten zu steuern, die mit einem auf Process Mining optimierten Harness (He et al., 2026) vollumfassend und autonom Analyseanfragen verarbeiten können. Die Agent Harness umfasst dabei Datenschichten für die operativen Systeme in Form von Model Context Protocol (MCP)-Servern sowie dedizierten Skills, um das optimale Vorgehen, die XES-Methodik sowie Validierungslogiken zu definieren. Mit weiteren, eingebetteten Tools sollten die Agenten die extrahierten Daten selbständig analysieren und fertige Berichte erstellen.

- **Vorteile:** Höchste Flexibilität; neue Systeme lassen sich durch Anbindung weiterer MCP-Server oder dedizierter Agenten integrieren, ohne, dass die unterliegende Pipeline angepasst werden muss. Der Agent kann unerwartete Datenstrukturen interpretieren.

- **Nachteile:** Die genutzten Daten enthalten typischerweise sensible Informationen, wie wichtige Unternehmenskennzahlen, aber auch personenbezogene Daten. Bei diesem Ansatz müssen einige juristische Schverhalte beachtet werden, darunter insbesondere die Datenschutz-Grundverordnung (DSGVO) (Europäisches Parlament und Rat der Europäischen Union, 2016) und den EU AI Act (Europäisches Parlament und Rat der Europäischen Union, 2026). Diese schreiben vor, dass KI-Systeme risikoklassifiziert und ordnungsgemäß doku-

mentiert werden.

Schließlich ist das Verhalten des Agenten nicht-deterministisch, was eine verlässliche Auswertung einschränkt. Auch die Qualität der Ausgabe variiert je nach Prompt und Formulierung. Man läuft Gefahr, dass der Agent falsche Analyseergebnisse liefert oder halluziniert. Bei diesem Ansatz ist eine kritische Hinterfragung der Ergebnisse durch einen Menschen unerlässlich.

2.1.2. Ansatz 2: Kooperativer Ansatz: Mensch und KI

Der zweite Ansatz kombiniert die Stärken der deterministischen ETL-Schritte mit gezielten KI-Aufrufen. Das Large Language Model (LLM) wird nur dort eingesetzt, wo es einen klaren Mehrwert bietet. Das ist insbesondere beim Schema Mapping der Fall. An anderen Stellen wiederum bieten sich oftmals statistische Mittel anstelle von KI an. Der Mensch bleibt hierbei der Verantwortliche. Die KI macht Vorschläge für beispielsweise Fremdverweise, Felder, Typen; der Mensch muss diese Vorschläge explizit akzeptieren. Auf Basis dieser Vorschläge laufen dann deterministische und damit reproduzierbare Auswertungen.

- **Vorteile:** Hohe Flexibilität; Das LLM kann dynamische, individuelle Datenstrukturen interpretieren. Mit dem hybriden Ansatz wird weiterhin eine Reproduzierbarkeit gewährleistet.

- **Nachteile:** Bei großen Datenmengen kann auch das Sichten und Übernehmen der Vorschläge sehr aufwendig werden. Da der Mensch hier aber gewollt der Verantwortliche bleiben soll, bietet es sich *nicht* an, diesen Schritt zu automatisieren. Auch hier muss sichergestellt sein, dass die Inferenz ge-

setzeskonform gemäß EU AI Act und DSGVO stattfindet.

2.1.3. Ansatz 3: Vollständiger Verzicht auf KI

Der dritte Ansatz kommt vollständig ohne LLM aus und setzt auf regelbasierte Erkennung (reguläre Ausdrücke, Heuristiken, Wörterbücher).

- **Vorteile:** Dieser Ansatz ist datenschutzrechtlich unkritisch, da kein Einsatzgebiet des EU AI Act greift. Zudem ist dieser Ansatz deterministisch und damit vollständig reproduzierbar.

- **Nachteile:** Die Erkennung individueller Datenstrukturen und der semantischen Rolle von Spalten ist mit diesem Ansatz nicht generalisierbar. Das Mapping der Schemata muss in jedem Projekt erneut durchgeführt werden, was zusätzlichen Aufwand bei den ohnehin schon eng budgetierten Projekten bedeutet. Das ist ein direkter Widerspruch zu der Zielsetzung dieses Projekts.

2.1.4. Entscheidung

Gewählt wurde **Ansatz 2**: ein kooperativer Ansatz, bei dem LLM-Aufrufe gezielt für Spaltenklassifikation und das Mapping der Schemata eingesetzt werden. Die Freigabe der Daten an das LLM erfolgt dabei nur auf Stichproben (wenige Tupel des Gesamtdatensatzes), wodurch das Datenschutzrisiko und die Antwortzeit minimiert werden. Die LLM-Antworten werden in einem lokalen Cache abgelegt, sodass identische Prompts nicht erneut gesendet werden, sondern stattdessen aus dem Cache gelesen werden. Der im Kapitel „Technische Umsetzung“ beschriebene

MCP-Server stellt eine Ergänzung zu dieser Strategie dar: Er stellt zusätzliche Metadaten der Schemata der Abas-ERP-Datenbank bereit und kann bei Bedarf weitere Felder und Tabellen für den Agenten zugänglich machen, ohne dass dieser direkten Vollzugriff auf die operativen Daten erhält.

2.2. Make-or-Buy

Neben der Eigenentwicklung wurde auch die Nutzung kommerzieller Process Mining Software in Betracht gezogen. Insbesondere *Celonis* sei bereits etabliert und biete einen dedizierten *Process Mining Connector* für die Datenanbindung an ERP-Systeme (Celonis SE, 2026). Dessen Lizenz- und Betriebskosten sind für mittelständische Kunden jedoch erheblich und bewegen sich damit außerhalb des wirtschaftlichen Rahmens, den die vorliegende Projektarbeit anstrebt. Zudem sind dessen proprietären Formate für eine wissenschaftliche Auswertung weniger geeignet, da sie die Überführung der Rohdaten in eine „Black Box“ verlagern und dem Anwender die Kontrolle über das Schema Mapping entziehen. Die Eigenentwicklung bietet demgegenüber volle Kontrolle über sämtliche Pipelineschritte, die Einhaltung datenschutzrechtlicher Vorgaben und eine vollständige Reproduzierbarkeit.

Die Validierung kommerzieller Software fand abseits dieser Projektarbeit durch einen anderen Personenkreis parallel statt und ist nicht Bestandteil der vorliegenden Projektarbeit.

2.3. Datenextraktion

Für die Extraktion der Rohdaten aus Abas ERP wurden drei Kerntechnologien untersucht.

2.3.1. edpexport.sh

Das mit Abas ERP ausgelieferte Shellskript **edpexport.sh** exportiert Datenbanktabellen als reine Comma-Separated Values (CSV)-Datei über die EDP-Schnittstelle. Ein Entscheidender Vorteil ist die vollständige Kompatibilität auch mit älteren Versionen von Abas ERP. Allerdings erzwingt das Skript eine Bedienung via Command Line Interface (CLI) und schreibt sämtliche Felder als reinen Text, inklusive variierender Datums- und Zahlenformate, die von der konfigurierten Bediensprache abhängen. Die so exportierten Rohdaten müssen von dem entfernten Server auf einen lokalen Server für die Analyse übertragen werden. Neben variierender Formate gibt es auch weitere proprietäre Datentypen, wie die Darstellung von Booleans als Werte „ja“ und „nein“ oder interne Verweisdarstellungen. Diese Typisierung erschwert die automatisierte Weiterverarbeitung erheblich.

2.3.2. SQL-Datenbanken

Abas ERP speichert seine Daten in einer proprietären Datenbank („LogDb“), die aufgrund des proprietären Protokolls kaum zugänglich ist. Ein indirekter Structured Query Language (SQL)-Zugriff wäre ausschließlich über eine vorgelagerte Schnittstelle möglich, was erheblichen Mehraufwand bedeutet. Aus akademischer Sicht

ist dieser Weg weniger interessant, da der Zugriff via EDP den kanonischen Kommunikationsweg mit Abas ERP darstellt.

2.3.3. Abas Middleware (REST-API)

Die Abas Middleware stellt eine REST-API mit einem spezifischen JavaScript Object Notation (JSON)-Format bereit. Sie liefert strukturierte Objektrepräsentationen unterteilt in Kopf- und Tabellenfeldern und ist allgemein über Hypertext Transfer Protocol (HTTP) erreichbar, was sie sehr zugänglich macht. Sie lässt sich damit auch direkt aus Python ansprechen. Die gelieferten Daten sind typisiert und verwenden kanonische Feldnamen, was die nachgelagerte Transformation vereinfacht. Demgegenüber steht, dass die Middleware eine authentifizierte Verbindung erfordert und eine relativ schlechte Geschwindigkeit der Datenabfragen aufweist.

2.4. Auswahl des passenden Datenformats

Für die Analyse von Prozessen gibt es zwei wesentliche Datenformate. Das *XES-Log-Format* ist ein in der Methodik des Process Mining etablierter Standard zum Speichern und Auswerten von Event Logs. Technisch ist ein XES-Log eine ausdefinierte eXtensible Markup Language (XML)-Datei. Diese Datei beinhaltet alle Daten zu einem bestimmten Prozess. Eine einzelne Aktivität (Prozessschritt) innerhalb des Logs wird *Event* genannt; werden mehrere Events zu einem Prozessdurchlauf gebündelt, spricht man von einem *Trace*.

Jedes Event hat mindestens die Attribute `Case-ID`, `Activity` und `Timestamp`.

Für weitere Analysemöglichkeiten können aber noch zusätzliche Attribute, wie die Kosten des Prozessschrittes oder die verwendeten Ressourcen, wie Mitarbeiter, gespeichert werden.

Das XES-Log ist case-zentriert aufgebaut. Das bedeutet, dass jedes Ereignis eine eigenständige, eindeutige Case-ID hat. Durch die Abbildung der Aktivitäten über die Zeitstempel lässt sich somit ein Prozessgraph rekonstruieren.

Unternehmensdaten und deren Beziehungen sind allerdings sehr vielfältig. So referenziert eine Bestellung beispielsweise Kunden, Artikel, Lagerplätze, Mitarbeiter, welche wiederum selbst in einer Vielzahl anderer Prozesse involviert sind. Das neuere *Object-Centric Event Log (OCEL)-Format* löst diese Problematik, indem es Events mit einer beliebigen Anzahl Objekte verknüpfen kann. Damit wird die starre Struktur des XES-Log aufgebrochen und es wird eine vollumfängliche Perspektive auf alle Objekte eröffnet.

Da das OCEL-Format bislang aber nicht sehr weit verbreitet und sowohl das Erstellen als auch die Analyse dieses Formats bisher komplex ist, wird vorerst das etablierte XES-Log-Format genutzt. Eine Übertragung in das OCEL-Format stellt aber eine mögliche Erweiterung des Projekts dar (siehe Kapitel „Ausblick“).

2.5. Technologien

Für die Technologien wurden im Wesentlichen zwei Ökosysteme in Betracht gezogen: Der Einsatz der Java Virtual Machine (JVM) ist im Umfeld Abas ERP sehr weit verbreitet und der Quasi-Standard für Entwicklungen innerhalb von Abas ERP. Demgegenüber steht die Verwendung von Python, welches den Quasi-

Standard für das Gebiet der Data Science darstellt.

2.5.1. Kotlin, Kotlin Notebooks, Kotlin DataFrame

Eine zunächst attraktive Option stellt die Verwendung von *Kotlin Notebooks* dar. Kotlin Notebooks reimplementieren den Notebook-Ansatz von Jupyter im Ökosystem der JVM. Das Entwicklungsframework AJO in Abas ERP ist vollständig kompatibel zur JVM und damit ebenso zur Programmiersprache Kotlin und den Kotlin Notebooks. Das eingebettete Kotlin Notebook bietet damit direkten Zugriff auf die operativen Daten des Abas ERP, strikte Typisierung und einen Laufzeitvorteil gegenüber der Abas ERP Middleware.

Es handelt sich dabei um eine junge Technologie. Da das Ökosystem für Process Mining in Python deutlich reifer ist und die Pipeline künftig auch weitere Systeme anbinden soll, stellt die Wahl von Kotlin Notebooks ein hohes Risiko dar. Darüber hinaus müssten viele bereits in Python vorhandene Logiken und Algorithmen für Process Mining und Auswertungen im Allgemeinen von Grund auf neu implementiert werden.

2.5.2. Python, marimo/Jupyter, pandas

Jupyter Notebooks sind der Quasi-Standard für Data Science Projekte in Python. Allerdings sind Jupyter Notebooks nur eingeschränkt versionierbar und reaktiv, das bedeutet, Zellen können in beliebiger Reihenfolge ausgeführt werden, was bei komplexen Pipelines zu subtilen Inkonsistenzen führt.

Die neuere Technologie *marimo (Marimo Team, 2026)* löst dieses Problem, indem sie den Quellcode des Notebooks als reines Pythonmodul speichert und eine reaktive Ausführungssemantik über Abhängigkeitsgraphen implementiert. Variablenreferenzen zwischen Zellen werden statisch analysiert; verändert der Anwender einen Parameter, werden alle abhängigen Zellen automatisch neu ausgeführt. Für die interaktive Bestimmung des LLM-Mappings und der Konformitätsschwellen ist dies ein erheblicher Produktivitätsgewinn. Ferner ist marimo bereits auf einen kooperativen Einsatz von KI ausgelegt, indem es die Sitzung laufender Notebooks einem Agenten verfügbar macht und vordefinierte Agent Skills liefert. Als Agent Harness wurde *OpenCode* gewählt. Dabei handelt es sich um eine quelloffene Software, die die wesentlichen Standards von Coding Agenten, wie Skills, Integrationen zu MCP-Servern und Tools, bereits vollständig implementiert.

Gewählt wurde letztlich der etablierte Ansatz über Python mit pandas und DataFrames. Als interaktive Oberfläche wurde jedoch marimo anstelle von Jupyter gewählt.

2.6. Ergebnisse der Ideenfindung

Der konkrete, erforschte Ansatz sieht eine Kooperation von Mensch und KI vor, um dynamische Schemata zu berücksichtigen und eine Entscheidungshilfe für das Mapping zu liefern. Dabei werden die Datenstrukturen passend zum zu analysierenden Prozess vorgeschlagen, gewählt und extrahiert. Ziel ist dabei ein Event Log im XES-Format, welches für weitreichende, abschließende Analysen, beispielsweise durch das Tool *ProM* (ProM Tools Consortium, 2026), genutzt wird.

Die Extraktion findet dabei mithilfe der Abas Middleware (REST-API) in der Pro-

grammiersprache Python statt. Softwarearchitektonisch wurde eine *Glass Box Architektur* gewählt. Am Ende jeder Phase wird ein versionierbares Artefakt generiert und kann eingesehen werden. Jeder Folgeschritt ist damit reproduzierbar. Die so extrahierten Daten werden dann in einem marimo Notebook auf einer Weboberfläche mithilfe eines DataFrame dargestellt. Durch ein dynamisches Mapping können vielfältige Prozesse verschiedener Unternehmen mit der gleichen Methodik abgebildet werden.

3. Herausforderungen

Während der Konzeptionierung und Implementierung der Pipeline traten mehrere, teils unerwartete Hürden auf, die die endgültige Architektur maßgeblich geprägt haben. Die folgenden Abschnitte dokumentieren diese im Detail.

3.1. Wahl des richtigen Sprachmodells

Eine der ersten und weitreichendsten Entscheidungen betraf die Wahl des Sprachmodells. Die im ERP-System vorgehaltenen Daten enthalten typischerweise personenbezogene Daten, womit die DSGVO und der EU AI Act greifen und eine Übermittlung an viele externe Inferenzanbieter von vornherein ausscheidet. Hinzu kommt die Verpflichtung einer Risikoklassifizierung für KI-Systeme (Europäisches Parlament und Rat der Europäischen Union, 2026) und eine Dokumentationspflicht.

Die Größe des Modells beeinflusst die Qualität der Mappingvorschläge erheblich. Kleine Modelle im Bereich von 7 bis 35 Milliarden Parametern lieferten selten gute Vorschläge und waren damit nicht nur unnützlich, sondern aktiv kontraproduktiv.

Für einen sinnvollen Einsatz kamen damit nur quelloffene Frontier Modelle infrage. Als Entscheidungsgrundlage dienten die Benchmarks von *lmarena.ai* (Arena Intelligence (formerly LMArena, UC Berkeley), 2026a). Diese lassen sich auch nach Open Source Modellen filtern. Datengrundlage der Benchmarks bildet die Anwendung in der echten Welt (real-world usage). Als neutraler Benchmarker und mit einem Datenbestand von ca. 900.000 Sitzungen bietet lmarena.ai eine hervorragende Datengrundlage für diese Entscheidung (Arena Intelligence (formerly LMArena, UC Berkeley), 2026b).

Benchmark quelloffener LLM Modelle:

Rang[1]	Modell	Entwickler	Lizenz	Net Imp.[2] (%)	Conf. Succ.[3] (%)	Sitzungen
10	GLM-5.2 (Max)	Z.ai	MIT	4,51	9,96	17.447
13	GLM-5.1	Z.ai	MIT	2,07	3,30	39.320
16	DeepSeek V4 Pro	DeepSeek	MIT	-0,76	-0,75	29.674
17	Kimi K2.6	Moonshot	modif. MIT	-1,01	-0,43	40.826
18	Kimi K2.7 Code	Moonshot	modif. MIT	-1,11	3,22	20.298
19	DeepSeek V4 Flash	DeepSeek	MIT	-1,70	4,35	39.458
24	Nemotron 3 Ultra	Nvidia	OpenMDW-1.1	-7,36	-5,54	5.733
25	Minimax M2.7	MiniMax	modif. MIT	-7,83	-12,51	39.324
27	Gemma 4 31B	Google	Apache 2.0	-12,72	-5,55	33.561

Tabelle 3.1.: Open Source Modelle aus dem Agent Arena Leaderboard von *lmarena.ai* (Arena Intelligence (formerly LMArena, UC Berkeley), 2026a) (Stand 22. Juni 2026, insgesamt 911.120 Sitzungen über 28 gelisteten Modelle).

[1] Rang im Gesamtranking über alle Modelle einschließlich proprietärer Modelle.

[2] Net Improvement: aggregierter Signal-Score der Erhebungsmethodik (Arena Intelligence (formerly LMArena, UC Berkeley), 2026b).

[3] Confirmed Success: Anteil der Sitzungen mit bestätigtem Erfolg.

Gewählt wurde letztlich das Modell *GLM-5.2* von *Z.ai* (Z.ai (zai-org), 2026). Die

Software wurde so gebaut, dass das Sprachmodell noch zu einem späteren Zeitpunkt per Konfiguration ausgetauscht werden kann.

3.2. Proprietäre Datenbank und proprietäre Protokolle

Abas ERP nutzt eine eigenentwickelte, proprietäre Datenbank. Diese speichert die Daten in einem Binärformat und verhindert selbst ein lokales Auslesen. Der einzige Zugriff auf die Datenbank findet über das explizit dafür vorgesehene EDP statt; die Kommunikation erfolgt dabei rein textbasiert über eine Socketverbindung.

Die direkte Socketverbindung via EDP wäre extrem aufwendig zu implementieren und damit unwirtschaftlich. Anstatt das native Protokoll direkt zu verwenden, muss daher eine Kompatibilitätsschicht, wie die Abas ERP Middleware genutzt werden. Das hat unter Anderem eine längere Laufzeit während der Datenextraktion zur Folge.

3.2.1. Abas ERP Middleware (REST-API)

Die Middleware kapselt das EDP und stellt eine REST-API bereit. Standardmäßig richtet sich diese nach dem Hypermedia as the Engine of Application State (HATEOAS)-Prinzip (Aydemir & Basciftci, 2022), das heißt zu jedem Datenobjekt werden zusätzlich Verknüpfungen in Form von Hyperlinks auf die entsprechende REST-Ressource ausgegeben, sowie eine Liste möglicher Operationen, die auf das referenzierte Objekt angewandt werden können. Das mag zwar praktisch wirken,

bedeutet allerdings einen deutlich größeren Payload für jedes Objekt. Das macht sich insbesondere bei der Laufzeit bemerkbar. Die Datenextraktion über die Middleware ist um ein Vielfaches langsamer als der Export via edpexport.sh. Glücklicherweise unterstützt die API mehrere Response-Formate. Über einen expliziten HTTP-Header im Request konnte ein deutlich leichtgewichtigeres Format angefordert werden. Abbildung 3.1 zeigt dieses Format.

```
{
  "resultData": [
    {
      "head": {
        "gruppe": 1,
        "grbez": "SalesOrder",
        "sn": "(164,0,0)"
      },
      "table": [
        {
          "zn": 1,
          "price": 50.0
        },
        [...]
      ]
    },
    [...]
  ]
}
```

Abbildung 3.1.: Beispielantwort einer Datensatzabfrage über die REST-API (vereinfacht)

Die Ergebnisse der Datenbankabfragen werden in Chunks geliefert. Damit müssen für jede Datenbankselektion mehrere Requests stattfinden („Pagination“ (Murali, Raj et al., 2019)), was die Laufzeit weiter erhöht. Zudem liefert die REST-API die Daten in einer Head-/Table-Struktur: Objekte bestehen aus einem Kopfobjekt und beziehen sich auf n Tabellenobjekte (Zeilen). Beim Aufbau des pandas DataFrame

mussten die Kopffelder dupliziert werden, sodass eine flache Tabelle entstand, die pro Tupel gesamtheitliche Daten enthält.

3.2.2. edpexport.sh

Das mit Abas ERP ausgelieferte Shellskript *edpexport.sh* ist der vom Hersteller vorgesehene Weg für die Massendatenverarbeitung. Es ruft EDP direkt auf und schreibt eine CSV-Datei. Im Sinne der Datenauswertung gibt es dabei allerdings folgende Probleme:

1. **Proprietäre Enkodierung:** Historisch bedingt verwendet Abas ERP einen eigens definierten Zeichensatz. Es gleicht dem Zeichensatz „Latin-1“. Dieses muss zuvor in einen modernen Standard, wie beispielsweise den 8-Bit UCS Transformation Format (UTF-8) Zeichensatz kodiert werden.

2. **Verschiedene Datumsformate:** In Abas ERP können verschiedene Bediensprachen und damit auch Datumsformate konfiguriert werden. Bei einem Datenexport werden die Datumsfelder damit als reiner String in verschiedenen Formaten ausgegeben.

3. **Verschiedene Zahlenformate:** In Abas ERP können verschiedene Bediensprachen und Zahlenformate konfiguriert werden. Bei einem Datenexport werden die Zahlen als Strings mit variierenden Dezimal- und Tausendertrennern und unterschiedlicher Anzahl an Nachkommastellen ausgegeben.

4. **Booleans:** Booleans innerhalb von Abas ERP werden nicht klassisch als „true“ oder „false“ ausgegeben, sondern als Strings „ja“ oder „nein“.

5. **Reservierte Zeichen:** Innerhalb der Abas Datenbank ist das Semikolon (;) ein reserviertes Zeichen für Zeilenumbrüche in mehrzeiligen Textfeldern. Gleichzeitig ist es aber auch der Feldtrenner für die ausgegebene CSV-Datei. Diese Problematik konnte umgangen werden, indem man in dem Aufruf des Shellskripts ein alternatives Zeichen für Zeilenumbrüche innerhalb der Felder definieren konnte (hier: „#“).

3.3. Dynamische Datenstrukturen

Die wohl größte Herausforderung liegt in der Dynamik der Datenstrukturen innerhalb von Abas ERP. Individuelle Variablen oder Datenbanken haben oftmals nur interne Kenner und sind kryptisch benannt. Sie enthalten nur selten offensichtliche Hinweise auf ihre semantische Bedeutung im Prozess. Unternehmensspezifische Prozesse oder Teilprozesse sind durch diese dynamischen Datenbanken fragmentiert. Eine nachträgliche Verknüpfung dieser Fragmente zu einem Prozessgraphen kann daher nicht nur rechen- und zeitintensiv sein, sondern erfordert oftmals auch einen gesamtheitlichen Blick über mehrere Datenbanken hinweg.

3.4. Keine konkreten Anforderungen

Zu Projektbeginn lagen kaum konkrete Anforderungen vor. Der Auftrag lautete vielmehr, die grundlegende Methodik für künftige Process Mining Projekte zu erforschen und zu vereinfachen. Das bedeutet einerseits architektonische Designfreiheit, andererseits mussten aber auch sämtliche Eventualitäten berücksichtigt werden, was sich letztlich stark auf den Aufwand auswirkte.

3.5. Zeitdruck

Das Projekt wurde mit fest definiertem Abgabetermin umgesetzt. Die parallele Evaluierung von mehreren KI-Strategien, Datenextraktionspfaden, Datenformaten und Frameworks stellte ein erhebliches Zeitrisiko dar. Die bewusste Entscheidung lieber eine schlankere aber vollständig getestete Lösung abzuliefern als ein überladenes System mit halbfertigen Features zog sich als Leitmotiv durch sämtliche Entscheidungen. So wurde beispielsweise das sehr passende OCEL-Format aus Zeitgründen in diesem Rahmen nicht umgesetzt und die tatsächliche Analyse der Prozesse auf externe Tools abseits des vorliegenden Projekts ausgelagert.

3.6. Halluzinationen

„[Halluzinationen beziehen] sich auf von einem KI-Modell generierte Inhalte, die zwar realistisch erscheinen, aber von den vorgegebenen Quelleninputs abweichen“ (Siebert, 2024).

Im ersten Wurf wurde angestrebt, das Schema Mapping und die Erkennung der Fremdverweise vollständig durch das LLM umzusetzen. Die Ergebnisse waren allerdings unbefriedigend. Nicht selten kam es vor, dass Beziehungen zwischen mehreren Tabellen nicht richtig erkannt oder halluziniert wurden. Die Folge waren fehlerhafte Verknüpfungen und letztlich ein Prozessmodell, das nicht der Realität entspricht.

Durch eine heuristische und statistische Auswertung (Kombination aus Werteüberlappung, Namens- und Typähnlichkeit) konnte diese Halluzination vermieden wer-

den. An anderen Stellen, in der KI zum Einsatz kommt, muss ein Mensch explizit generierte Vorschläge genehmigen, was dieser Symptomatik aktiv entgegenwirkt.

4. Technische Umsetzung

Dieses Kapitel beschreibt die konkrete, technische Implementierung der zuvor definierten Process Mining Methodik. Es werden System- und Softwarearchitektur, der Technologie Stack sowie der Programmablauf beschrieben. Da die Nutzung von KI eine besondere Bedeutung in diesem Projekt hat, wird auch in besonderem Maße auf das Agent Framework[1] und die generelle Konfiguration des Agenten eingegangen.

4.1. Eingesetzte Technologien

Technologie	**Einsatzgebiet**
Python	etablierte Programmiersprache für Data Science
pandas	Datenmanipulation und Datensichtung
marimo	Interaktive, KI-unterstützte Notebooks
pm4py	Process-Mining-Bibliothek für XES Event Logs
httpx	HTTP-Client für Abas-REST-API und LLM-Anbindung
OpenCode	LLM Anbindung (Mapping der Schemata und Konformitätsprüfung)
Abas Middleware	Primärer Weg der Datenextraktion aus Abas ERP
ProM	Externes Werkzeug für die Prozessanalyse

Tabelle 4.1.: Technologieentscheidungen (Technologie Stack)

[1] Ein *Agent Framework* beschreibt die Gesamtheit der für die Erzeugung eines KI-Agenten eingesetzten Technologien

4.2. Systemarchitektur

Abas ERP ist selbstgehostete Software und wird auf einem einzelnen Server installiert. Dieser Server nutzt eine aktuelle Version des Linux-Derivats RHEL. Die Middleware ist aus dem kundeninternen Netzwerk erreichbar. Die Graphical User Interface (GUI) von Abas ERP kann allerdings ausschließlich von Windows Rechnern ausgeführt werden. Die Kommunikation findet dabei auch innerhalb des Netzwerks serverübergreifend via EDP statt.

Aus Latenzgründen wurde das marimo Notebook direkt auf dem Linux Server eingerichtet, auf dem auch lokal das Abas ERP läuft. Über die marimo Weboberfläche konnten die Notebooks auch von einem Windows Rechner im gleichen Netzwerk gesteuert und entwickelt werden.

4.3. Softwarearchitektur

Die gesamte Methodik folgt der Maxime *volle Transparenz durch ausdefinierte Schritte*. Dabei ist die Pipeline in mehrere Phasen unterteilt, die jeweils ein klares Ergebnis in Form eines Artefakts produzieren. Diese Artefakte sind die Basis für die Folgephasen. Damit sind die Phasen reproduzierbar und einzeln anpassbar. Insbesondere in Kombination mit nicht-deterministischen Programmen, wie es mit KI der Fall ist, ist eine solche softwarearchitektonische Grundlage essentiell. Diese Architektur nennt man *Glass Box Architektur*.

Das marimo Notebook dient als Control Plane, in dem die Ergebnisse jeder Phase dargestellt und editiert werden können, ohne die entwickelten Module im Hinter-

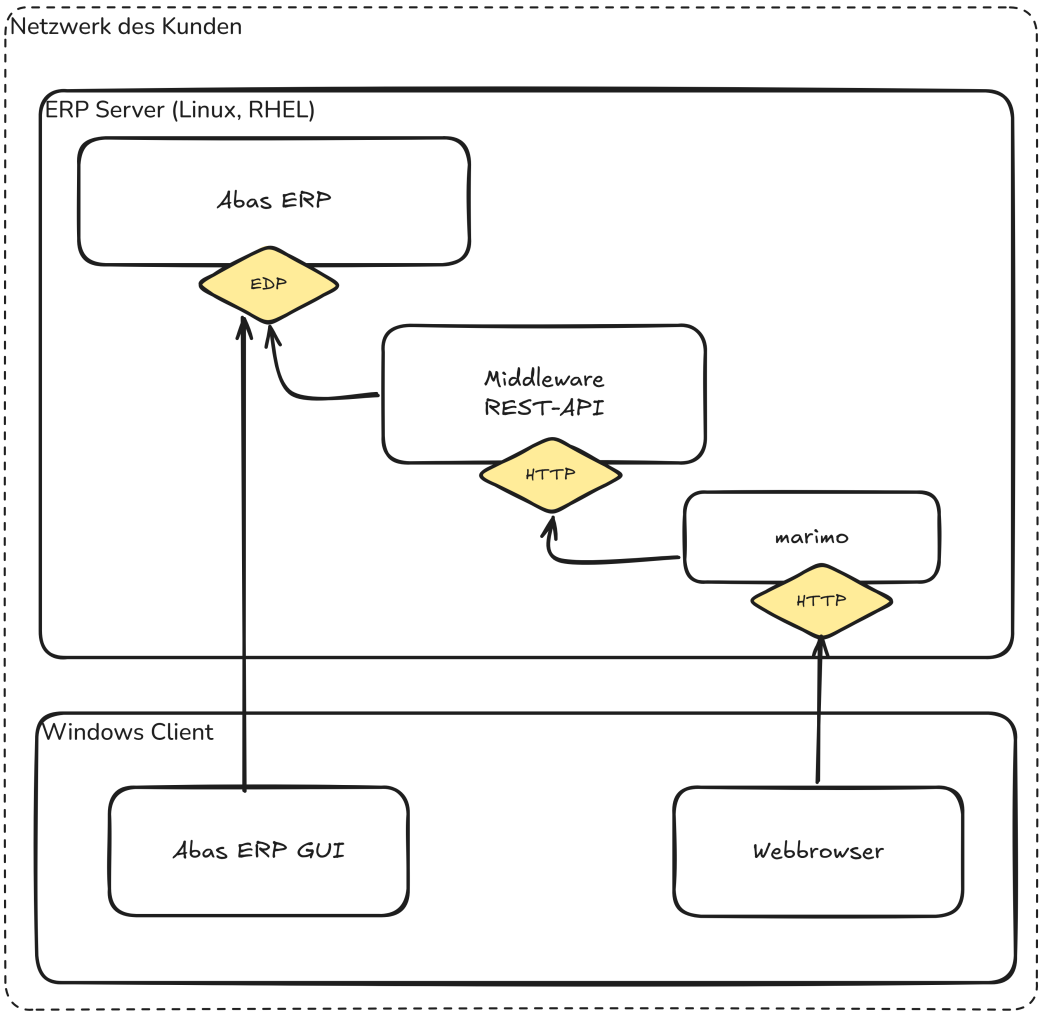

Abbildung 4.1.: Schaubild der Systemarchitektur

grund (Quellcode) verändern zu müssen.

4.4. Agent Framework

Ein Agent Framework ist die Infrastruktur zum Ausführen, Steuern und Orchestrieren autonomer KI-Agenten. Dabei vereint es softwareseitig etablierte Standards und Methodiken im Umgang mit Sprachmodellen. Insbesondere die Möglichkeit des „function calling“ (OpenAI, 2025) und die Nutzung von „agent skills“ (Agent Skills Initiative, 2026) haben die Autonomie und Kapazitäten von KI-Agenten extrem gesteigert.

Das quelloffene Agent Framework *opencode* (SST & OpenCode Contributors, 2026) erleichtert dabei nicht nur die Nutzung dieser Standards, sondern kann zugleich über eine REST-API direkt in die Process Mining Pipeline angebunden werden.

Durch die Befähigung des KI-Agenten kann auf dem marimo Notebook autonom iteriert werden. Der KI-Agent erkennt dynamische Datenstrukturen und Felder selbstständig und kann bei Bedarf weitere Felder für die nachfolgende Pipeline anbinden.

Insgesamt verfügt der Agent über das notwendige Wissen und die notwendigen Tools, um die vordefinierte Methodik nachzustellen. Damit kann man, sofern notwendig, den KI-Agenten für zusätzliche Iterationen oder tiefere Analysen über das marimo Notebook hinaus konsultieren.

4.4.1. Skills

Wesentlicher Bestandteil für einen verlässlichen Agenten sind Skills. Im Rahmen dieser Projektarbeit wurden mehrere Skills genutzt:

marimo-Skill: marimo bietet umfangreiche Unterstützung für Integrationen mit KI-Agenten an. Durch vordefinierte Skripte und einem dedizierten Skill zur Nutzung ebendieser kann ein Agent sich mit laufenden Instanzen eines marimo Notebook verbinden, Variablen auslesen und sogar Zellen bearbeiten. Zudem bietet marimo einen eingebauten KI-Chat, sodass ein Sprachmodell zusätzlich Fragen rund um das laufende Notebook beantworten kann.

Process Mining Skill und XES Log Skill: Es wurden zudem dedizierte Skills für diese konkrete Process Mining Methodik implementiert. Darin wird das XES-Log-Format erläutert, Referenzen mitgeliefert und Skripte zum Validieren der Logs geliefert. Zudem wurden in weiteren Skills die konkreten Schritte des KI Agenten erläutert und ein Rahmen sowie Ziel für die Aufgaben des Agenten definiert.

Skills für Abas ERP: Letztlich wurden noch Skills geschaffen, die den Umgang mit der Abas ERP Middleware definieren und die allgemeine Datenstruktur sowie individuelle Datentypen erläutern.

4.4.2. MCP-Server

Über einen dedizierten MCP-Server erhält der KI-Agent direkten Zugriff auf die Datenbank von Abas ERP und kann sowohl die zugrundeliegenden Datenbanken, als auch die darin liegenden Felder inklusive Datentypen und Feldbeschreibungen

abrufen.

4.5. Programmablauf

Die Pipeline besteht aus vier aufeinander aufbauenden Phasen, die jeweils ein numeriertes Unterverzeichnis von `data/` befüllen:

4.5.1. Phase 1: Datenextraktion und Ingestion

Die Daten werden hier bevorzugt via REST-API extrahiert und in den Arbeitsspeicher (DataFrame) geladen. Als Fallback kann hier auch ein Import einer CSV-Datei erfolgen, wie beispielsweise zuvor via edpexport.sh exportiert wurde. Vorgesehen ist, dass damit auch Daten weiterer Fremdsysteme geladen werden.

Artefakte

`data/01_ingestion/_extraction_report.json` gibt eine Übersicht, wie viele Daten in den DataFrame geladen wurden, welche Spalten es gibt, welche Datentypen diese haben und einen ersten Überblick, wie konsistent die Daten sind (null-Werte). Das geladene DataFrame kann zusätzlich im marimo Notebook gesichtet werden.

`data/01_ingestion/table_recommendation_cache.json` enthält generierte Empfehlungen (0..10) des Agents, welche Datenquellen (Tabellen) für die folgende Pipeline genutzt werden sollen. Als konkrete Felder gibt es (1) einen Tabellenkenner

(`table_spec`), (2) die Relevanz als Ganzzahl zwischen 0 und 10 (`relevance_score`), (3) einen Begründungstext, wieso die Datenquelle relevant ist (`reasoning`), sowie (4) konkrete Feldvorschläge (`recommended_head_fields` und `recommended_table_fields`).

4.5.2. Phase 2: Schema Mapping

In dieser Phase werden die extrahierten Felder klassifiziert für die Weiterverarbeitung in das XES-Log. Dabei wird für die Zielfelder `case_id`, `activity`, `timestamp`, `resource` und `custom` jeweils eine Liste von gewichteten Vorschlägen erzeugt. Jeder Vorschlag besitzt auch hier wieder die Felder `confidence_score` und `confidence_note` als Begründung des Agenten für die Einstufung.

Ferner werden hier auch Vorschläge für Fremdschlüsselbeziehungen erstellt. Diese beziehen sich hierbei noch ausschließlich auf die gesamte Tabelle: z.B. wird erkannt, dass eine Tabelle „Auftrag“ einen grundsätzlichen Verweis auf die Tabelle „Verkaufsposition“ haben könnte. Auch diese haben wieder eine Konfidenzbewertung inklusive Begründung. Das konkrete Feldmapping findet in der nächsten Phase mit statistischen Mitteln statt.

In dieser Phase muss der Anwender interaktiv die Vorschläge akzeptieren oder ablehnen. Er hat auch die Möglichkeit, Tabellen zu wählen, die nicht durch den Agenten vorbelegt werden.

Artefakte

Das Ergebnis dieser Phase ist das Artefakt `data/02_mapped/_mapping_report.json`. Diese Datei speichert die vom Nutzer endgültig gewählten Tabellen und

Feldmappings. Auf Basis der verknüpften Tabellen findet in der nächsten Phase eine Konkretisierung durch ein Mapping auf Spaltenebene statt.

Zudem findet ein Caching jeglicher Interaktion mit dem Agenten statt und wird in dem Artefakt `data/02_mapped/_llm_cache.json` persistiert. Dadurch kann man nachträglich in diese Phase zurückzukehren, ohne erneut den Agenten aufrufen zu müssen. Es macht diese Phase zudem reproduzierbar.

4.5.3. Phase 3: Relationale Verarbeitung

In dieser Phase werden die Beziehungen zwischen den Tabellen konkretisiert. Durch eine statistische Auswertung über eine Kombination aus Namensheuristik und Werteüberlappung werden auch hier zu den aus Phase 2 gewählten Tabellen Verknüpfungsvorschläge der zuvor festgelegten `case_id` erstellt. Bei einem hohen Score werden diese auf der Oberfläche des Notebook vorbelegt. Auch hier muss der Anwender die Vorschläge explizit übernehmen, hat aber auch die Möglichkeit, Felder zu wählen, die nicht vorbelegt/ermittelt wurden.

Artefakte

Ergebnis des Mappings ist das Artefakt `data/03_relational/_relational_report.json`. Dieses enthält Metainformationen, wie die Anzahl verknüpfter Tabellen und die Anzahl verknüpfter Spalten. Es enthält aber auch die tatsächlichen in der Oberfläche gewählten Mappings in der Form `<tabelle>.<spalte> -> <tabelle>.<spalte>`, z.B. `Auftrag.warenempf -> Kunde.nummer`

4.5.4. Phase 4: Generierung des XES-Log

In dieser Phase wird letztlich deterministisch das XES-Log gebaut. Dazu wird die Bibliothek *pm4py* (Berti et al., 2023) genutzt. Die zuvor definierten Spalten werden in das von pm4py vorgeschriebene Format gebracht, z.B. das Feld `concept:name`, welches zuvor `activity` hieß. Letztlich wird über die Bibliothek das konkrete XES-Log auf eine Datei geschrieben. Im marimo Notebook wird eine Directly-Follows Graph (DFG) gerendert; dieser dient aber nur einer schnellen Übersicht und keiner tiefgehenden Analyse. Ab diesem Punkt kommen dedizierte externe Tools, wie *ProM* für die weitere Prozessanalyse zum Einsatz. Peter T.G. Hornix beschreibt dieses Vorgehen ausführlich im Rahmen einer Masterthesis (vgl. Hornix, 2007).

Artefakte

Das Artefakt dieser Phase ist eine zum XES-Standard konforme XML-Datei. Diese Datei dient als Grundlage für alle nachfolgenden Schritte der Methodik, wie etwa eine Konformitätsprüfung oder einer Root-Cause-Analyse. Die Pipeline und das marimo Notebook enden mit der Generierung des XES-Log als Basis für weitere Analysen.

5. Ergebnisse

Die gesamte Datenpipeline über alle vier Phasen hinweg, als auch der kooperative Einsatz von KI bewährten sich. Das Ergebnis des Projekts ist ein dynamisch generiertes XES-Log, welches zur weiteren, tiefen Analyse genutzt werden kann.

Als Anwendungsfall wurde exemplarisch ein Prozess aus dem Einkaufs- und Beschaffungsumfeld des Kunden analysiert. Eine detaillierte Beschreibung des konkreten Prozesses sowie der zugehörigen Kennzahlen ist aus Vertraulichkeitsgründen nicht Bestandteil dieser Ausarbeitung.

5.1. Einsatz von KI

Der gewählte kooperative Ansatz hat sich im Rahmen dieser Projektarbeit bewährt. Sowohl das gewählte Sprachmodell (*GLM-5.2*), als auch das Agent Framework erwiesen sich als nützlich in der Erzeugung nachvollziehbarer Vorschläge. Das war insbesondere bei individuellen Datenstrukturen, die ansonsten hohen Einarbeitungsaufwand erfordert hätten, ein großer Mehrwert. Auch die Entscheidung, die Vorschläge explizit auf der Weboberfläche des Notebooks freigeben zu lassen, erwies sich als korrekt. Nicht zuletzt durch das Caching des Agenten trugen diese

Entscheidungen zur Reproduzierbarkeit und zur Transparenz der Pipeline bei.

5.2. Datenextraktion und Ingestion

Die Datenextraktion wurde flexibel umgesetzt, indem zwei funktionierende Wege für die Datenbeschaffung realisiert wurden. Dabei war die Abfrage via REST-API die primäre Methode. Ein Fallback über einen CSV-Export wurde ebenso implementiert.

Dabei gab es grundsätzlich keine allgemein bessere Extraktionsmethode. Der Export via REST-API lieferte eine sauberere Datengrundlage auf Kosten der Laufzeit. Der Massenexport als CSV-Datei war für große Datenmengen geeignet, lieferte allerdings eine geringere Datenqualität.

5.3. XES-Log

Das Endziel der Pipeline wurde erreicht. Ein dynamisch generiertes und reproduzierbares XES-Log dient als Datengrundlage für eine weitreichende Prozessanalyse über externe, dedizierte Tools abseits der Pipeline.

5.4. Lessons Learned

Während der Konzeptionierung und Implementierung ergaben sich wesentliche Erkenntnisse für künftige Data Science Projekte. Der Einsatz künstlicher Intelligenz

erwies sich als herausfordernd, führte aber letztlich zu sehr guten Ergebnissen. Entscheidend dabei ist nicht nur die Wahl des korrekten Sprachmodells, sondern auch in besonderem Maße die richtige Gestaltung und Implementierung der Agentenlaufzeit (Agent Framework). Darunter gehört die Wahl der richtigen Tools und MCP-Server, aber auch Skills und die Ausgestaltung des Agenten in Form der richtigen System- und Userprompts. Auch die Wahl der Glass-Box-Architektur war ein voller Erfolg und lieferte die unabdingbare Transparenz.

5.5. Nicht erreichte Ziele

Aufgrund zeitlicher Restriktionen konnten die anfangs definierten Ziele und Anforderungen nicht vollständig umgesetzt werden. So wurde zwar die Implementierung des XES-Logs umgesetzt; das OCEL-Log-Format wurde aber nicht implementiert. Auch eine genauere, wirtschaftliche Gegenüberstellung dieser Projektmethodik und kommerzieller Alternativen im Sinne einer Make-or-Buy-Entscheidung wurde im Rahmen dieser Projektarbeit nicht durchgeführt. Letztlich wurde auch die abschließende Analyse des XES-Logs auf externe Tools (zum Beispiel *ProM*) ausgelagert und ist somit nicht Bestandteil dieser Ausarbeitung.

6. Ausblick

6.1. Make-or-Buy-Entscheidung

Sekundäres Ziel dieser Projektarbeit war es, eine Entscheidungsgrundlage für die Make-or-Buy-Entscheidung zu schaffen. Die in dieser Arbeit erforschte Process Mining Methodik wird folgend kommerziellen Lösungen wie Celonis gegenübergestellt, um letztlich eine Entscheidung zu treffen, ob diese Methodik in weiteren Kundenprojekten eingesetzt wird.

6.2. Implementierung des OCEL-Logs

Großes Potenzial liegt in der Umsetzung des OCEL-Log-Formats. Insbesondere im ERP-Umfeld sind die Datenobjekte und Prozesse eng miteinander verwoben. Ein Ansichtswechsel auf Objektebene ermöglicht dabei tiefergehende Analysen und offenbart Muster, die mit dem XES-Log nicht einfach ersichtlich wären.

6.3. Anbindung weiterer Fremdsysteme

Bisher sind die Daten auf das ERP-System beschränkt. Das bedeutet jedoch, dass Prozesse nicht über Systemgrenzen und Schnittstellen hinweg ersichtlich sind. Künftig sollen auch weitere Systeme, wie beispielsweise Customer Relationship Management (CRM)- oder PLM-Systeme in die Pipeline integriert werden, um einen vollumfänglichen Blick über Systemgrenzen hinweg zu gewinnen.

A. Abkürzungsverzeichnis

KI Künstliche Intelligenz

ERP Enterprise Resource Planning

PLM Product Lifecycle Management

IT Informationstechnologie

ETL Extract, Transform, Load

OCEL Object-Centric Event Log

XES eXtensible Event Stream

RHEL Red Hat Enterprise Linux

EDP ERP Datenübertragungsprotokoll

TCP Transmission Control Protocol

RAM Random Access Memory

CPU Central Processing Unit

AJO Abas Java Objects

REST Representational State Transfer

API Application Programming Interface

FOPs Flexible Oberflächenprogramme

LLM Large Language Model

MCP Model Context Protocol

CSV Comma-Separated Values

SQL Structured Query Language

HTTP Hypertext Transfer Protocol

JSON JavaScript Object Notation

BPM Business Process Management

DSGVO Datenschutz-Grundverordnung

CRM Customer Relationship Management

HATEOAS Hypermedia as the Engine of Application State

UTF-8 8-Bit UCS Transformation Format

AI Artificial Intelligence

CLI Command Line Interface

DFG Directly-Follows Graph

GUI Graphical User Interface

JVM Java Virtual Machine

XML eXtensible Markup Language

KPIs Key Performance Indicators

B. Tabellenverzeichnis

C. Abbildungsverzeichnis

D. Literaturverzeichnis

Agent Skills Initiative. (2026). Agent Skills – Documentation [Standardisierung von Agent Skills für KI-Coding-Agenten]. Verfügbar 20. Juni 2026 unter https://agentskills.io/home

Arena Intelligence (formerly LMArena, UC Berkeley). (2026a). Arena Leaderboard: Open Source Model Rankings [Crowdsourced LLM leaderboard; open-source model rankings, Stand 22. Juni 2026]. Verfügbar 22. Juni 2026 unter https://lmarena.ai/leaderboard

Arena Intelligence (formerly LMArena, UC Berkeley). (2026b). How Arena Works: Crowdsourced AI Model Evaluation Methodology [Beschreibung der Erhebungsmethodik (Battle Mode, paarweiser Vergleich, Net-Improvement-Score)]. Verfügbar 22. Juni 2026 unter https://lmarena.ai/how-it-works

Aydemir, F., & Basciftci, F. (2022). Application of HATEOAS principle in RESTful API design. *2022 IEEE 22nd International Symposium on Computational Intelligence and Informatics and 8th IEEE International Conference on Recent Achievements in Mechatronics, Automation, Computer Science and Robotics (CINTI-MACRo)*, 000051–000056.

Berti, A., van Zelst, S., & Schuster, D. (2023). PM4Py: A process mining library for Python. *Software Impacts*, *17*, 100556. https://doi.org/10.1016/j.simpa.2023.100556

Celonis SE. (2026). Celonis Process Mining: Execution Management System [Celonis Process Mining and Execution Management Platform]. Verfügbar 20. Juni 2026 unter https://www.celonis.com/process-mining

Europäisches Parlament und Rat der Europäischen Union. (2016). Verordnung (EU) 2016/679 (Datenschutz-Grundverordnung, DSGVO) [Amtsblatt der Europäischen Union, L 119/1]. Verfügbar 22. Juni 2026 unter https://eur-lex.europa.eu/eli/reg/2016/679/oj?locale=de

Europäisches Parlament und Rat der Europäischen Union. (2026). Regulation (EU) 2024/1689 – Artificial Intelligence Act (AI Act Explorer) [Offizielle interaktive Fassung der EU-Verordnung über künstliche Intelligenz]. Verfügbar 22. Juni 2026 unter https://artificialintelligenceact.eu/ai-act-explorer/

Forterro Deutschland Abas GmbH. (2026). Abas ERP – Software für Fertigungsunternehmen im Mittelstand | Forterro [Offizielle Produktseite von Abas ERP]. Verfügbar 14. Juni 2026 unter https://abas-erp.com/de

He, C., Zhou, X., Wang, D., Xu, H., Liu, W., & Miao, C. (2026). Harness engineering for language agents: The harness layer as control, agency, and runtime [Konzeptpapier zur Harness-Softwarearchitektur für Sprachmodell-Agenten].

Hornix, P. T. (2007). Performance analysis of business processes through process mining. *Master's Thesis, Eindhoven University of Technology*.

MAIT Gruppe. (2026). MAIT – Ihr Digitalisierungspartner. Auf Augenhöhe. Wegweisend. [Offizielle Website der MAIT-Gruppe]. Verfügbar 14. Juni 2026 unter https://mait-group.com/

Marimo Team. (2026). Marimo: A Reactive Python Notebook for Experiments, ML, and Data Apps [Open-source reactive Python notebook framework]. Verfügbar 20. Juni 2026 unter https://marimo.io/

Murali, A., Raj, P., et al. (2019). *Hands-On RESTful API Design Patterns and Best Practices: Design, develop, and deploy highly adaptable, scalable, and secure RESTful web APIs*. Packt Publishing Ltd.

OpenAI. (2025). Function Calling (API Guide) [Offizielle Dokumentation zu Function Calling der OpenAI-API]. Verfügbar 20. Juni 2026 unter https://developers.openai.com/api/docs/guides/function-calling

ProM Tools Consortium. (2026). ProM – Process Mining Tools [Open-Source Werkzeugsuite zur Prozessanalyse und Prozesskonformitätsprüfung]. Verfügbar 20. Juni 2026 unter https://promtools.org/

Red Hat, Inc. (2026). Red Hat Enterprise Linux [Offizielle Produktseite zu Red Hat Enterprise Linux]. Verfügbar 14. Juni 2026 unter https://www.redhat.com/de/technologies/linux-platforms/enterprise-linux

Siebert, J. (2024). Halluzinationen von generativer KI und großen Sprachmodellen (LLMs) [Blogpost zum Phänomen der Halluzinationen generativer KI]. Verfügbar 22. Juni 2026 unter https://www.iese.fraunhofer.de/blog/halluzinationen-generative-ki-llm/

SST & OpenCode Contributors. (2026). OpenCode: An Open-Source AI Coding Agent [Open-Source Coding-Agent mit MCP- und Skills-Unterstützung]. Verfügbar 20. Juni 2026 unter https://opencode.ai/

Z.ai (zai-org). (2026). GLM-5.2 Model Card [Flagship-Modell für Long-Horizon-Tasks, 1M-Token-Kontext, MIT-Lizenz]. Verfügbar 22. Juni 2026 unter https://huggingface.co/zai-org/GLM-5.2

Eidesstattliche Erklärung

Studierender: Manuel Ott
Matrikelnummer: 907778

Hiermit erkläre ich, dass ich diese Arbeit selbstständig abgefasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

.....	.....
Ort, Abgabedatum	Unterschrift (Vor- und Zuname)

wbh
WILHELM BÜCHNER
HOCHSCHULE