SPIE RODIAS

ETL-Programm zur automatisierten JSON-Datenintegration in eine relationale Datenbank

Dokumentation zur betrieblichen Projektarbeit

IHK-Abschlussprüfung Winter 2025/2026

Fachinformatiker für Anwendungsentwicklung

Auszubildender: Sven Kanter, Bremserstraße 128, 67063 Ludwigshafen

Bildungsträger: Comcave.College GmbH, Rheinuferstraße 9, 67061 Ludwigshafen

Betrieb: SPIE RODIAS GmbH, Eisleber Straße 4, 69469 Weinheim

Betreuer: Oliver Kemmer, Uwe Müller

SPIE Germany Switzerland Austria

SPIE INDUSTRY SERVICE & WIND

SPIE RODIAS GmbH

Eisleber Straße 4

69469Weinheim

SPIE RODIAS GmbH

Geschäftsführer:
PeterStängle

sales@rodias.de

+49-6201-503-0

www.spie-rodias.de

Sitz der Gesellschaft: Weinheim

Registergericht: Amtsgericht Mannheim
HRB-Nr.: 736005

USt-IdNr.: DE 132496149

Sparkasse Erlangen

IBAN: DE40 7635 0000 0000 0241 10

BIC: BYLADEM1ERH

SPIE RODIAS

Änderungshistorie:

Datum	Änderung durch	Was wurde geändert
24.10.2025	Sven Kanter	Neuerstellung initial, Korrekturen

Autor: Sven Kanter

Version: 1.0

Datum: 24.10.2025

SPIE Germany Switzerland Austria
SPIE INDUSTRY SERVICE & WIND

SPIE RODIAS GmbH
Eisleber Straße 4
69469 Weinheim

SPIE RODIAS GmbH
Geschäftsführer:
André Panné

sales@rodias.de
+49-6201-503-0

www.spie-rodias.de

Sitz der Gesellschaft: Weinheim
Registergericht: Amtsgericht Mannheim
HRB-Nr.: 736005
USt-IdNr.: DE 132496149

Sparkasse Erlangen
IBAN: DE40 7635 0000 0000 0241 10
BIC: BYLADEM1ERH

Inhalt

1 Einleitung

1.1 Ausgangssituation

Die SPIE RODIAS ist ein Unternehmen, das sich seit mehr als 35 Jahren als Spezialist für die Digitalisierung der Instandhaltung komplexer technischer Anlagen etabliert hat. Ihre Expertise liegt darin, Versorgern dabei zu helfen, interne Prozesse zur Leistungserbringung zu digitalisieren sowie den sicheren Rückbau von Anlagenkapazitäten zu unterstützen. Insbesondere in der Nuklear-Branche im deutschsprachigen Raum, hat sich RODIAS als führend positioniert.

Unser Kunde verwendet das von RODIAS, vor rund 30 Jahren hergestellte, Instandhaltungs-EAM in seiner aktuellen Version auch „OPEN TALOS“ genannt.

Dieses spezielle EAM ist ein sehr komplexes Client-Server System auf dem sich mehrere hundert autorisierte Anwender, mit Peaks zu Revisionszeiten, gleichzeitig bewegen um mit dem EAM - geplante, jederzeit behördlich überprüfbare Instandhaltung zu betreiben.

Unser Kunde verwendet zur Modellierung, zum Betrieb und zur behördlichen Überprüfbarkeit relevanter und komplexer Instandhaltungsprozesse das EAM OPEN TALOS.

Eine auf dem Konzept der Petri Netze basierte Ablauf-Engine rechnete diese modellierten Geschäftsprozesse und delegiert die anstehenden Tätigkeiten (Vorgänge, Aktivitäten) an die Zuständigen Fachabteilungen zur Bearbeitung.

Die Ablauf-Engine benötigt einen, im OPEN TALOS, erstellten, modellierten, hinsichtlich der Ablauf-Engine konsistenten und freigegebenen modellierten Geschäftsprozess.

Hierzu bietet das OT, einen für Softwareentwickler konzipierten, Editor an mit 2d graphischer Oberfläche. Ansonsten liegen die Ablauf Daten, sowohl auf Applikationsseite wie auch auf Modellseite, rein tabellarisch vor.

Die Zustandsdaten eines Ablaufs werden zum Zweck der Versionierung, in einer JSON-Datei hierarchisch gespeichert und in eine SVN basierte Versionskontrolle überführt.

Ein direkter Zugriff auf die entsprechenden Daten in der Instandhaltungsdatenbank ist verboten. Unter anderem weil das EAM OT die Datenbank komplett selbst verwalten muss, damit die bisher sehr geschätzte Zuverlässigkeit des EAMs sichergestellt bleibt.

Die Dateien zur Versionierung (GWF-Dateien) der Geschäftsprozesse dürfen zum Dokumentationszweck weiterverarbeitet werden.

An dieser Stelle setzt dieses Projekt an und es soll eine Softwarelösung entwickelt werden, welche die notwendigen Daten relational bereitstellt, damit weitere Applikationen auf die, im Wesentlichen, Graphen (Knotenbeschreibung und Kantenbeschreibung) Daten - verlässlich zugreifen können.

Einzelne Ablaufpläne können ein bis ca. hundert Knoten und entsprechend viele Kanten enthalten. Eine Umgebung kann ca. 100 – 200 modellierte Ablaufpläne enthalten unterteilt in Hauptnetzte und Teilnetze.

Bisher wurde die Problematik das nur die Softwareentwicklung des OT EAMs eine Diagrammansicht des Geschäftsprozessgraphen hat so gelöst das händisch das entsprechende Ablaufdiagramm nachgezeichnet wurde oder es musste ein Entwickler Workspace mit vielen Abhängigkeiten bereitgestellt werden, um eine Diagrammansicht des Ablaufplans anzuzeigen und auch diese Diagrammansichten sind ohne Standard und im Vergleich zu UML eher wild gezeichnete Aktivitätsdiagramme aber auch technisch konsistent zu der Ablauf Engine des OT und fachlich, gemäß den Anforderungen, über Jahrzehnte hinweg angepasst.

1.2 Projektziel

1.2.1 Allgemeines Projektziel

Das Allgemeine Projektziel, Diagramme im Standard BPMN 2.0, automatisiert auf Basis der OT GWF-Dateien zu erzeugen wird komplementär verfolgt.

Ziel dieses Projekts ist die Erstellung eines leichtgewichtigen ETL-Programms, welches genau nur die benötigten Daten aus den GWF-Dateien des OT EAMs extrahiert (extract), die extrahierten Graphen-Daten optional gemäß den Programmparametern modifiziert (transform) und diese Daten in eine relationale Datenbank zuordenbar speichert (load).

1.2.2 Funktionale Anforderungen

Am Mittwoch, 08.10.2025 fand eine letzte Abstimmungsrunde mit dem Kunden und dessen Softwareentwicklung statt. Die hier beschriebenen Anforderungen basieren auf dieser letzten Abstimmung.

- Das ETL-Programm soll in Java17 entwickelt werden. Damit dieses [Programm] kompatibel mit einem V24 OT Workspace ist und in der gemeinsamen SVN-Versionskontrolle als Java-Projekt mitgeführt werden kann. OT V24 wird 2025 ausgerollt.
- Das ETL-Programm soll unabhängig von OT betrieben werden können. Ein ausgecheckter SVN-Stand mit der entsprechenden Verzeichnisstruktur und den darin enthaltenen GWF-Dateien ist notwendig. Ein laufender OT-Server soll jedoch nicht notwendig sein.
- Das ETL-Programm soll als Batch-Job ausgeführt werden können, zum Beispiel via Cronjob, Aufgabenplaner oder vergleichbaren Scheduler, um regelmäßig automatisiert die Datenbank zu aktualisieren.
- Das ETL-Programm soll auf mehrere Umgebungen ausgerichtet werden können da es von einem OT-System mehrere sequenziell bediente Umgebungen gibt

  (Entwicklungsumgebung -> Testumgebung -> Abnahme-Umgebung -> Qualitätsumgebung -> Produktiv-Umgebung)
- Ebenso soll das ETL-Programm als Data-Loader betrieben werden können und nur einzelne selektierte GWF-Dateien sollen pointiert in die relationale Datenbank gespeichert werden können.
- Das ETL-Programm soll konfigurierbar mittels Programmparameter (CLI) gem. den Anforderungen sein.
- Das ETL-Programm soll mit einem System zur File Auswahl, Suche und Filterung welches gem. den Anforderungen, die relevanten Files ermittelt – ausgestattet sein***. [Änderung gem. Besprechung am 08.10.2025: Es sollen alle Ablaufplandaten extrahiert werden, nicht nur diese im Status: RELEASED, UPDATED -> weniger Aufwand]***
- Ein spezieller Parser soll anhand der gelieferten Pfade, die Files ermitteln und genau nur die notwendigen Attribute aus der JSON-Hierarchie der GWF-Datei entnehmen, welche für die Weiterverarbeitung der Daten notwendig sind.
- Eine Typisierung der Geschäftsprozess-Graphen Daten ist vorzunehmen um darauf aufbauend die rein technischen Knoten aus den Graphen Daten der Ablauf-Engine des OT zu identifizieren, zu entfernen und zu überbrücken***. [Änderung gem. Besprechung am 08.10.2025:***

Die Überbrückung der Knoten soll mittels Programmparameter optional erfolgen können- > mehr Aufwand]

- Die bereinigten Daten der Ablaufpläne (Geschäftsprozesse) sind relational zu Speichern.

  ER-Modell und Datenbankschema sind zu erstellen.

- DAO und DM-Klassen zur Umsetzung der CRUD-Methoden sind zu implementieren
- Die Administration der Datenbank erfolgt durch den Kunden. ***[Änderung gem. Besprechung am 08.10.2025: Eine Konsole zur Einsicht und Administration der Datenbank wie in der H2 Datenbank bereitgestellt ist ausreichend da vorerst nur im Fehlerfall auf die Datenbank administrativ zugegriffen werden muss]***
- Zur Qualitätskontrolle soll ein abschließendes Exportprotokoll je Durchlauf in Form eines Log Files erstellt werden, um eine reibungslose Datenbereitstellung zu dokumentieren oder im Fehlerfall bei der Analysierung von Fehlern behilflich zu sein.

1.2.3 Nicht-Funktionale Anforderungen

- **Leistungsanforderungen – gering**

  Generell sind die Anforderungen an die Performance gering jedoch sollte ein Durchlauf der Datenbereitstellung, je ca. 100 modellierte Geschäftsprozessplan-Daten, in unter 5 Sekunden erfolgen.

- **Sicherheitsanforderungen - gering**

  Die Sicherheitsanforderungen sind, in diesem Fall, gering. Das Programm wird in den Energieanlagen intern und von der Außenwelt abgeschirmt von den IT-Fachkräften vor Ort eingerichtet und betrieben. Somit ist ein unberechtigter Zugriff von außen ausgeschlossen da die Sicherheitsvorkehrungen der Energieanlage greifen. Die Daten verlassen die IT-Infrastruktur des Kunden nicht.

- **Zuverlässigkeit und Verfügbarkeit - mittel**

  Die Schnittstelle zur Datenbereitstellung wäre nicht kritisch für den Betrieb. Auch ein mehrwöchiger Ausfall der Datenbereitstellung kann nach der Fehlerbehebung schnell wieder auf den neusten Stand aufgearbeitet werden.

- **Wartbarkeit und Erweiterbarkeit - hoch**

  Das ETL-Programm ist für den dauerhaften zyklischen Betrieb angedacht. Versionsänderungen seitens des OT oder Anforderungen von geplanten Applikationen zum Beispiel werden ein stetiges Anpassen des ETL-Programms erfordern. Die Dokumentation wird durch Unittests im Grey-Box Verfahren gestützt und bietet somit eine zuverlässige Grundlage für Erweiterungen.

- **Portabilität und Kompatibilität - gering**

  Das ETL-Programm muss auf dem Betriebssysteme Windows10/11 ausführbar sein.

  Durch die konsequente Umsetzung in Java wäre ein Betrieb des Programms unter Linux unproblematisch jedoch ist vom Kunde ein Client-Seitiger Betrieb vorgesehen und sämtliche Clients des Kunden nutzen das Betriebssystem Windows10 oder Windows11.

  Weitere Kompatibilitäten sind bisher nicht bekannt.

- **Benutzbarkeit - mittel**

  Das ETL-Programm muss möglichst einfach zu konfigurieren sein mit genau nur den angeforderten Optionen. Durch Bereitstellung einer Bedienungsanleitung soll die Inbetriebnahme von einer IT-Fachkraft zügig (15min) vorgenommen werden können.

- **Skalierbarkeit - gering**

  Die Anforderungen an die Skalierbarkeit sind bisher gering. Durch die relationale Speicherung der bereitgestellten Ablaufplan Daten (Geschäftsprozesse) wird jedoch eine gewisse Grundskalierbarkeit gegeben. Die Datenbank kann in größere relationale Datenbanken integriert werden.

1.3 Projektschnittstellen

Die GWF-Dateien werden in der SVN-Versionskontrolle gespeichert. Ein laufender OT-Server ist zwar nicht notwendig jedoch muss ein vollständiger Workspace aus dem SVN ausgecheckt werden damit valide GWF-Dateien aus der Verzeichnisstruktur eines OT-Workspace ermittelt werden können.

Rücksprachen mit der Framework Entwicklung des OT waren hier notwendig gerade bei der Ermittlung der relevanten Attribute eines Ablaufplanes, der Petri-Netz Funktionalität und ganz allgemein zum grundsätzlichen Verständnis des Autors über den OT-Kontext im Allgemeinen.

Aufgrund der Weiterverarbeitung der Daten in der relationalen Datenbank in weiteren geplanten Applikationen war eine Abstimmung, Anpassung und Abnahme des Datenbankschemas notwendig. Diese wurde von dem Projektbetreuer des Autors in Abstimmung mit der Entwicklung des Kunden getätigt.

Alle weiteren Tätigkeiten des Projekts wurden von dem Autor selbständig durchgeführt.

1.4 Projektabgrenzung

Dieses Projekt extrahiert, transformiert und persistiert die Ablaufplan-Daten zuordenbar zu einem OT-Geschäftsprozess.

Aufgrund des Umfangs wird eine Konvertierung des jeweiligen Graphen in das BPMN 2.0 XML-Format in einem Folgeprojekt erledigt und ist **nicht** Bestandteil dieses Projekts.

Des Weiteren werden die Ablaufplan-Daten nur zum Zweck der Visualisierung in Diagrammen extrahiert und nicht um in einer anderen Ablauf-Engine, in einem anderen Format (BPMN2.0 XML) genauso prozessiert werden zu können. Hierzu wäre ein wesentlich umfangreicheres Projekt notwendig, um das Verhalten der OT Petri-Netz Engine ganz genau in einer anderen Engine abzubilden.

2 Projektablauf

Der Ablauf dieses Abschlussprojektes, das durch den Autor im Rahmen seiner Abschlussprüfung zum Fachinformatiker mit der Fachrichtung Anwendungsentwicklung durchgeführt wurde, wird im Folgenden strukturiert von dem Autor erläutert.

2.1 Information

Erste E-Mail-Korrespondenzen mit dem Kunden zur Problematik der „Workflow Visualisierung“ stammen aus dem Jahr 2020. Das OT EAM speichert sogenannte GWF-Dateien (Auszug GWF-

Datei im Anhang) mit den kompletten Ablaufplandaten erst ab der V20, welche vor rund 4 Jahren in Betrieb genommen wurde.

Die Anforderungen und das Angebot wurden vom RODIAS Consulting, der Projektleitung und dem Projektbetreuer in einem Workshop und drei Abstimmungsterminen unter anderem ermittelt. Der Autor war Protokoll führend bei zwei Terminen dabei.

Nachdem das Angebot vom Kunden angenommen wurde, begann die eigentliche Projektplanung.

Die Projektplanung des ersten Teilprojekts, die Erstellung des ETL-Programms, wurde vom Autor übernommen.

Ein, durch das V-Modell erweitertes, Dokumentgetriebenes Wasserfallprojekt, eingeteilt in sequenzielle Phasen, einer groben vorab Ressourcenplanung sowie qualitätssichernder Maßnahmen (Protokoll, Tests und Abnahmen) während des Entwicklungsprozesses, wurde vom Autor ausgewählt.

2.2 Ressourcen

Hardware: BAP- Berufsarbeitsplatz mit Laptop und 2 Monitoren

Sprachen: Java 17, SQL, Windows-Batch-Skript

Definitionen: JSON, Graphentheorie, Petri-Netze

Bibliotheken: JUnit5, GSON, java17JDK

Software:

- Eclipse IDE zur Entwicklung
- Eclipse IDE mit Plugin „eclipse4openTalos“ zur Server Kommunikation und Ergebnisüberprüfung
- Versionskontrolle: SVN geteilt mit dem Kunden
- EPF-Konfigurationsdatei für die Eclipse-IDE damit im SVN die Stände verglichen werden können (gleiche Code Formatierung mit dem Kunden)
- SonarLint (Code Formatierung, Code Smells)
- Tempo und Dokumentation: Atlassian Jira
- Diagramm Editor Draw.io
- MS Office (Word, Excel, PowerPoint)
- MS Teams für Rückfragen, Abstimmungen, Kundentermine
- Windows 11

Datenbank: H2

Kontext: OPEN TALOS Instandhaltung-EAM, BPMN 2.0 XML

Personal:

- Betreuer: Review des Codes, Datenbankschema Abnahme
- Auszubildender: Umsetzung des Projekts
- Kunde: Anforderungsermittlung und Abnahme

- Projektleitung: Anforderungsermittlung mit dem Kunden und Angebotserstellung

2.3 Analyse

2.3.1 Analyse-Phase

Die Analyse Phase wurde grob auf 12h geschätzt.

Der Autor analysierte die finalen Anforderungen und arbeitete aus den Anforderungen an das Gesamtprojekt die Anforderungen für dieses erste Teilprojekt heraus. Danach wurden die GWF-Dateien (Auszug GWF-Datei im Anhang) analysiert und die benötigten Attribute ermittelt. Rücksprachen mit dem Betreuer und der Framework Entwicklung fanden zu einzelnen Attributen statt. Abschließend wurden die Ressourcen festgelegt und dokumentiert.

Die grobe Schätzung ging in der Analyse-Phase gut auf.

2.3.2 Planung-/Entwurf-Phase

Die Planung und Entwurf-Phase wurde grob auf 28h geschätzt.

Zuerst skizzierte der Autor das System zur Abgrenzung des Hauptprojekts zu diesem Teilprojekt und ermittelte die Anwendungsfälle, um die Aufgaben des ETL-Programms aufzuzeigen.

Danach wurde die Programmarchitektur mittels eines Klassendiagramms (siehe Anhang Klassendiagramm) geplant mit allen beteiligten Datenstrukturen. An dieser Stelle waren die Ergebnisse aus der Analyse-Phase sehr effektiv, um das Klassendiagramm zügig zu ermitteln.

Ebenso wurde eine ENUM-Datenstruktur zur Typisierung der Knotenobjekte während des Parsens entworfen.

Graphen Daten Knoten Typen des OT-EAM:

- Start-Knoten
- Initial-Knoten
- Vorgangsknoten (M, MC: Muß-Vorgänge)
- Vorgangsknoten (O, OC: Kann-Vorgänge)
- Haltepunkte
- Entscheidungspunkte
- Teilnetzknoten
- Endknoten
- Return-Knoten
- Programmatisch-Knoten
- Konnektor
- Trigger-Knoten

Danach wurden die RECORD-Datenstrukturen zum Speichern der gesamten benötigten Ablaufplan-Daten entworfen und in das Klassendiagramm integriert.

In einem Aktivitätsdiagramm (siehe Anhang) wurde der Algorithmus zum optionalen Reduzieren der Knoten gemäß den Anforderungen entworfen.

Danach wurde das ER-Modell und das Datenbankschema erstellt und die DAO und DM-Klassen entworfen.

Am Ende dieses Tages fand eine Abnahme des Datenbankschemas durch den Betreuer statt.

1. Es sollen Surrogaten Schlüssel bei allen Tabellen verwendet werden.
2. In den relationalen Daten lassen sich 3 Gültigkeitsbereiche verschiedener IDs abgrenzen.
    - Tabellen-ID: eindeutig im Bereich der DB der Primärschlüssel
    - Model-ID: eindeutig auf der jeweiligen OT-Umgebung
    - Graph-ID: eindeutig innerhalb des jeweiligen Graphen

Um die Lesbarkeit zu erhöhen, wurden bessere Attributes Bezeichnungen gefunden.

Anschließend wurde das System zum Parsen der Programmparameter entworfen und die Zeilen der Protokolldatei definiert.

Die Planung und Entwurf-Phase endete 4h früher wie geschätzt. Der Autor entschied sich nicht sofort mit der Implementierung fortzufahren, sondern eine Überprüfung der Entwürfe vorzunehmen. Hierbei konnten Planungsfehler, fehlende Klassen etc. vom Autor entdeckt und korrigiert werden.

2.4 Umsetzung

Die Umsetzung erfolgte im Test Driven Development Verfahren. Hierbei wurde die jeweilige Testmethode vom Autor zuerst geschrieben, danach die aktiven Methoden zur Befriedigung des Tests und gefolgt von Refrakturierungen des Quellcodes gemäß den Prinzipien Lesbarkeit, Einfachheit, Redundanzfreiheit, Klassen und Methoden folgen dem Eine-Aufgabe Prinzip und der Testbarkeit.

Die Testabdeckung umfasste hauptsächlich die Anweisungsüberdeckung. In der Zeit konnte keine 100% prozentige Anweisungsabdeckung erreicht werden.

Dieses Vorgehen wird oft als Grey-Box Test bezeichnet, weil unter anderem zum Zeitpunkt der Testerstellung der aktive Quellcode nur potenziell vorliegt.

2.4.1 Test Implementierung Refrakturierung

Die Phase Test-Implementierung-Refrakturierung wurde grob auf 32h geschätzt.

Zuerst wurden die Testklassen angelegt und die ersten Tests auf Existenz durch Erstellung der jeweiligen Klasse befriedigt.

Danach wurde vom Autor die „FileWalker“ Klasse implementiert damit zu einem angegebenen Lese Pfad alle validen GWF-Dateien ermittelt werden. Darauffolgend wurde dann die „ProgramArgsParser“ Klasse erstellt, welche die Programmparameter aufbereitet und die daraus resultierenden Optionen zur weiteren Verwendung bereitstellt.

Mit diesen beiden fertigen, getesteten Klassen konnte der Autor den Parser und die geplante Knotentypisierung implementieren mitsamt den zugehörigen RECORD-Datenstrukturen, welche Hierarchisch gemäß der Hierarchie der GWF-Datei erstellt wurden damit ein einfaches Ergänzen, um weitere Attribute erleichternd ermöglicht wird.

Dann wurde die Klasse „NodesRemoverByType“ erstellt mit der Aufgabe die reinen Graphen-Daten zu modifizieren. Die Knoten des Typs: Konnektor werden immer überbrückt (siehe Aktivitätsdiagramm im Anhang). Die Knotentypen: Programmatik, Trigger und Kann-Vorgänge werden nur bei entsprechenden Programmparametern überbrückt (siehe auch optionale Knotentypen im Anhang).

An dieser Stelle wurde es eng mit der groben Schätzung da zwei perfide Fehlwirkungen beim Einlese Test aller Abläufe einer OT-Umgebung sich zeigten und die Behebung der Ursachen hatte weitere Stunden beansprucht.

Mit etwas Verzögerung ging es weiter mit der Entwicklung der Database Access Objects und der Datenmodellklassen.

Am Ende eines intensiven Tages, auch hier war die grobe Schätzung ein wenig zu gering, konnten die Tests zu den CRUD-Methoden erfolgreich befriedet werden und die ersten fehlerfreien Durchläufe (siehe Anhang) fanden statt. Die Datenbank wird angelegt, wenn es noch keine gibt, Die Tabellen ebenso und die Datensätze werden erstellt, wenn es zu dem jeweiligen Ablauf noch keine Datenbankeinträge gibt, ansonsten wird der Datensatz aktualisiert.

Abschließend wurde die „Log“ Klasse integriert und die Protokolldatei Erstellung gemäß der Planung umgesetzt.

Die Log Klasse wurde vom Autor basierend auf „java.util.logging“ implementiert jedoch nicht während des Projektzeitraums. Die Funktionalität, die einzelnen geloggten Zeilen in einen Puffer zu schreiben und am Ende eines Durchlaufs eine Protokolldatei mit Zeitstempel zu erstellen, wurde jedoch in der Projektzeit vom Autor implementiert.

2.5 Abnahme

2.5.1 Erstellung der Dokumentation

Die Erstellung der Dokumentation wurde grob auf 8h geschätzt.

Während des gesamten Entwicklungsprozesses wurde die Software: „Atlassian Jira“ zur Dokumentation verwendet. Dadurch konnte die Dokumentation in 4h, zügiger wie geplant zusammengestellt werden.

2.5.2 Abnahme Datenbankschema

Während der Planung und Entwurf-Phase wurde durch den Betreuer das ER-Modell sowie das Datenbankschema abgenommen. Dabei wurde u.a. die Verwendung von Surrogaten Schlüssel vorgeschrieben.

2.5.3 Bereitstellung und Abnahme durch Kunde

Die Abnahme des ETL-Programms erfolgte durch den Betreuer mit dem Kunden in einem ca. 1-stündigen Termin. Die umgesetzten Anforderungen wurden vom Autor kurz demonstriert.

Die Bereitstellung erfolgte zum einen für Entwickler im geteilten SVN-Repository und zur Verwendung wurden im Kundendownloadbereich ein Bereitstellungs-Zip Datei hinterlegt.

gwf2h2.zip

- gwf2h2.jar
- start.bat

- start_admin_konsole.bat
- bedienungsanleitung_gwf2h2_v1-0.pdf
- test [Ordner]
    - ressourcen

2.5.4 Benutzer-/Entwicklerhandbuch

Der Auslieferung liegt eine Bedienungsanleitung bei: „bedienungsanleitung_gwf2h2_v1-0.pdf“.

Die Bedienungsanleitung ist Adressatengerecht an die äußerst fachkundigen Systemadministratoren und Softwareentwicklerinnen des Kunden gerichtet und fällt somit sehr knapp und präzise aus.

Zum Beispiel musste der Autor den Adressaten nicht erklären wie man Windows-Batch-Skript verwendet und eine start.bat Datei erstellt jedoch wie die mitgelieferte, beispielhafte: „start.bat“ Datei effektiv an die lokalen Erfordernisse anpasst und nutzt, um verschiedene OT-Umgebungen anzusteuern.

Aufgrund der Fülle an Personen bezogener Daten, welche unter den Datenschutz fallen (DSGVO und BDSG) aber auch aus Gründen der Informationssicherheit (Umgebungspfade, Interna über die Struktur des OT-EAM) und dem damit verbundenen Aufwand eine zensierte Version der Bedienungsanleitung zu erstellen entschied sich der Autor an dieser Stelle nur zensierte Auszüge aus der Bedienungsanleitung zu präsentieren.

Auszüge aus der Bedienungsanleitung:

2.5.4.1 Inhalt Bedienungsanleitung

2.5.4.2 Auszug Screenshots

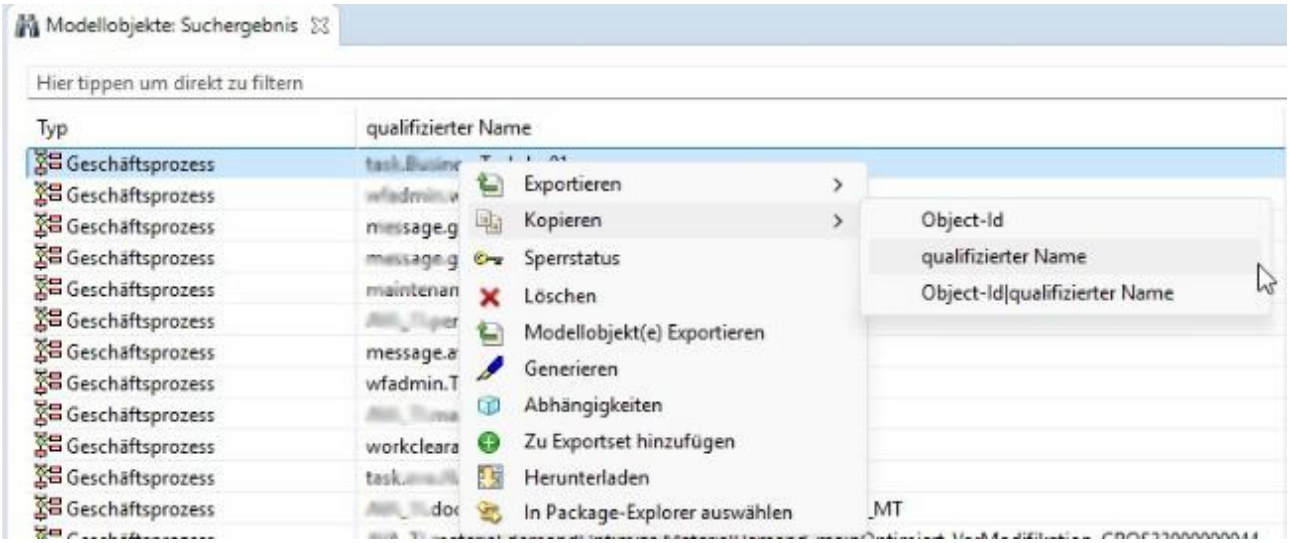


Abbildung 1: Auszug 3.2 vollqualifizierten Namen ermitteln (unkenntlich)

Abbildung 2: Beschreibung der "start.bat" Datei aus der Bedienungsanleitung (unkenntlich)

3 Retrospektive

3.1 Wirtschaftliche Betrachtung

Das ETL-Programm wurde mit überschaubarem Ressourceneinsatz realisiert: Entwicklung und Test erfolgten auf einem einzelnen Arbeitsplatz mit vorhandenen Tools (Eclipse, SVN, H2), was die direkten Projektkosten primär auf Arbeitszeit und Lizenzfrei-Software begrenzte. Durch die Entscheidung für Java 17 und eine leichtgewichtige H2-Datenbank konnte die Implementierung portabel und

wartungsarm gehalten werden, wodurch Folgeaufwände für Infrastruktur gering bleiben. Auch liegt der komplette Quellcode bis auf wenige Abhängigkeiten vor und kann somit erweitert getestet und speziell weiterentwickelt werden.

3.2 Ausblick und gelernte Lektionen

3.2.1 Ausblick

- Kurzfristig: Stabilisierung der Auslieferung (kleinere Fehlerbehebungen), Ergänzung der Bedienungsanleitung mit erweiterten Beispielen und FAQs gemäß dem Kundenfeedback, sowie ggf. vereinfachte Start-Skripte für verschiedene OT-Umgebungen.
- Kurzfristig: Aufgrund kurzfristiger Anforderungsänderungen die Option konfigurieren zu können, technische Knoten zu überbrücken zusammen mit dem Wunsch genau nur eine Version eines Ablaufplan zu persistieren ist konfliktbehaftet da in einem Durchlauf alle Graphen-Knoten persistiert werden und in einem anderen Durchlauf möglicherweise eine andere Konstellation – je nach gewählter Option. Dieser Umstand wurde bei der kurzfristigen Anforderungsänderung vom Autor aber auch seitens des Kunden und Consulting nicht beachtet. Hierzu sollte eine klar definierte Lösung angestrebt werden zum Beispiel durch Einführung eines „Format“ Attributes, welches persistiert wird und kennzeichnet auf welche Art und Weise der jeweilige Ablaufplan extrahiert, modifiziert und persistiert wurde.
- Mittelfristig: Implementierung eines optionalen BPMN-2.0-Exporters als Folgeprojekt, damit die erzeugten relationalen Daten direkt in Standard-BPMN-2.0-Editoren übernommen werden können.
- Langfristig: Optionale Migration des: „Backends“ auf ein produktives RDBMS (z. B. PostgreSQL) und Ergänzung um Web-basierte Administrations- und Visualisierungswerkzeuge, falls der Kunde eine zentralisierte Nutzung vorsieht.
- Langfristig: Realisierung, durch die OPEN TALOS Produktentwicklung, einer offiziellen OT-Schnittstelle. Diese wäre jedoch sehr aufwendig und von einem einzelnen Kunden in der Regel nicht wirtschaftlich vertretbar umzusetzen. Es gibt jedoch einen „Roundtable“ der OT-Kunden in dem solche Vorhaben gemeinsam geplant, beschlossen und beauftragt werden können.

3.2.2 Gelernte Lektionen

- Planung schützt vor Implementierungsfallen: Die zusätzliche Prüfphase nach Entwurf reduzierte Nacharbeit und verhinderte Architekturfehler; dies sollte als Standard im Projektplan beibehalten werden.
- TDD und testgetriebene Entwicklung lohnen sich: Frühe Tests der Klasse: „FileWalker“, Parameterparser und DAO vereinfachten die Fehlersuche und gaben Sicherheit bei Refrakturierungen. Tests sollten in zukünftigen Releases weiter ausgebaut werden, um allgemein die Testabdeckung zu erhöhen und erweiterte Test wie zum Beispiel Integrationstests darauf aufbauend zu erstellen.
- Dokumentation zielgruppengerecht halten: Die knappe, technische Bedienungsanleitung war für Systemadministratoren und die Softwareentwicklung des Kunden passend und dient

mehr wie ein Entwicklerhandbuch zum Nachschlagen. Sollten sich die Anwenderkreise erweitern wäre eine detaillierte Anleitung und ggf. eine Schulung notwendig.

- Konfigurierbarkeit erhöht die Komplexität: CLI-Parameter und die Option, technische Knoten zu überbrücken, erhöhten die Nützlichkeit erheblich. Bei künftigen Anforderungen sollten Konfigurationsoptionen modular und gut versionier bar gestaltet werden.
- Kurzfristige Anforderungsänderungen in letzter Minute müssen ganz besonders erörtert werden:

  In der letzten Abstimmungsrunde kurz vor dem Projekt Begin wurde durch den Kunden die Anforderung: „nur bestimmte Ablaufpläne welche sich im Status: UPDATED oder RELEASED befinden (siehe Zustandsdiagramm in der Anlage) zu extrahieren“; geändert.
  Es sollten alle Ablaufpläne ohne Beachtung des Status extrahiert werden. Dadurch konnte das eindeutige Attribut: „modeId“ nicht mehr als UNIQUE gekennzeichnet werden, denn bei Abläufen ohne entsprechende Status bekommen anstelle der Model ID eine Zeichenkettenkonstante geschrieben. Alternative wurde dann der „voll qualifizierte Name“ eines Ablaufplan als eindeutige Spalte gefunden.

4 Abkürzungsverzeichnis

4.1 SPIE

Mutterkonzern der SPIE RODIAS GmbH

4.2 RODIAS

SPIE RODIAS GmbH: Hersteller und Dienstleister des OPEN TALOS EAM.

4.3 OT

OPEN TALOS: komplexes Client-Server Instandhaltungs-EAM. OT erfüllt u.a. die Kriterien der Datensicherheit: Vertraulichkeit, Integrität, Verfügbarkeit und Authentizität und befindet sich derzeit im Zertifizierungsprozess der ISO 270001. Personenbezogenen Daten werden auf den Entwicklungsumgebungen pseudoanonymisiert, d. h. anhand der Daten kann kein Personen Bezug mehr hergestellt werden. Eine Umkehrung der Anonymisierung ist nicht möglich.

4.4 EAM

Enterprise Asset Management

4.5 Petri-Netz

Diskrete Modelle vorwiegend verteilter Systeme. Wikipedia

4.6 Geschäftsprozess

Rechenbarer Petri-Netz basierter, modellierter Plan eines Ablaufs im OPEN TALOS EAM

4.7 SVN

SVN (Subversion) ist ein zentrales Versionskontrollsystem

4.8 ETL

Programmmuster: Extract, Transform, Load

4.9 Batch-Job

regelmäßiger kurzzeitiger Hintergrundprozess, nicht interaktiv

4.10 Data-Loader

kurze Ausführungszeit (1-5 Sekunden), Einzelextraktionen

4.11 CLI

Command Line Interface (Kommandozeilen-Interface)

4.12 H2

leichtgewichtiges, in Java geschriebenes relationales Open-Source-Datenbankmanagementsystem

4.13 GWF-Dateien

Dateien mit der Dateiendung: „.gwf“ werden von dem OT EAM abgelegt und enthalten im JSON-Format alle Daten zu einem Ablaufplan.

4.14 OT Workspace

Ein Workspace welcher das Modellieren und Programmieren des EAM-Modells ermöglicht. Ausgestattet mit ca. 250-300 API und Microservices.

4.15 Surrogaten Schlüssel

Wenn in den originalen Datensätzen kein eindeutiger Primärschlüssel vorhanden und die Bildung eines Primärschlüssels durch mehrere Spalten nicht gewünscht wird dann kann zum Datenimport in die relationale Datenbank eine ID-Spalte erstellt und automatisch inkrementiert werden. Dieser neue Schlüssel wird auch Stellvertreter-Schlüssel oder auch Surrogaten Schlüssel genannt da dieser in den Quell Daten nicht vorkommt.

5 Anhänge

5.1 Detaillierte Zeitplanung

Phase: Analyse (12h)

Analyse des finalen Lastenhefts	2h
Analyse der EAM-Dateien, relevante Attribute ermitteln	4h
Erstellung Pflichtenheft, Ressourcenplanung	6h

Phase: Planung/Entwurf (28)

Systementwurf, High-Level Sequenzdiagramm	2h
Klassendiagramm erstellen	4h
Knotentypisierung der Graphen Daten ausarbeiten, Algorithmus zum Reduzieren der Graphen Daten entwerfen und entsprechendes Sequenzdiagramm erstellen	4h
ER-Modell und Datenbankschema erstellen CRUD-Methoden planen	8h
CLI entwerfen, Filterregeln erstellen, Lesealgorithmus entwerfen	8h
Protokollierungsdatei strukturieren	2h

Phase: Test und Refrakturierung (16h)

Grundaufbau der Testsuite erstellen	1h
Unittests erstellen (gem. TDD im Wechsel mit Implementierung und Refrakturierung) Testerstellung und Refrakturierung sind ungefähr auf die gleiche Dauer der entsprechenden Implementierungen geschätzt.	15h

Phase: Implementierung (16h)

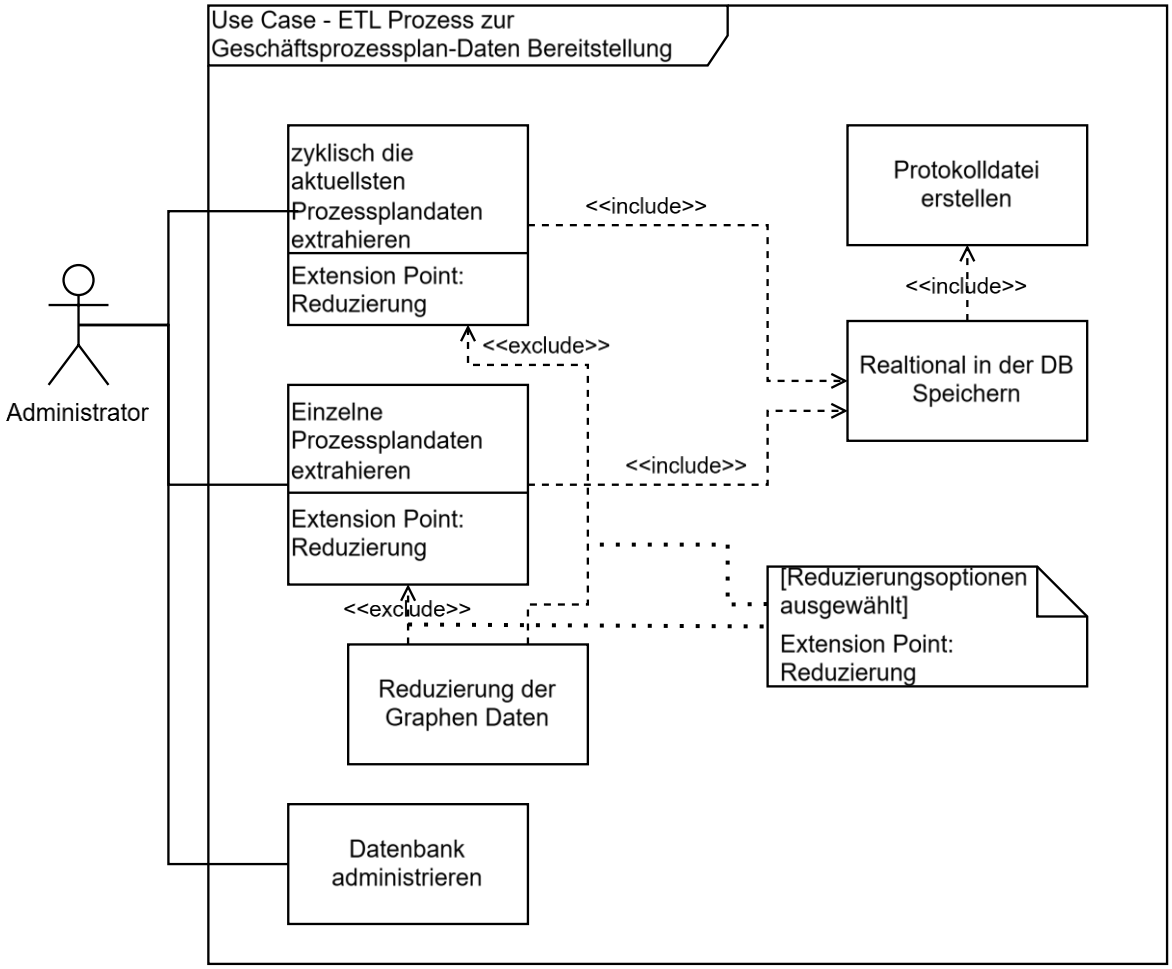
Klassen und weitere Datenstrukturen anlegen	1h
Implementierung CLI und Lesealgorithmus	2h

Implementierung des Parsers und der Typisierung	3h
Implementierung Modifikation der Graphen Daten	4h
Implementierung des Datenbankmanagements (DAO und DM-Klassen)	4h
Implementierung der Protokollierungsdatei Erstellung	2h

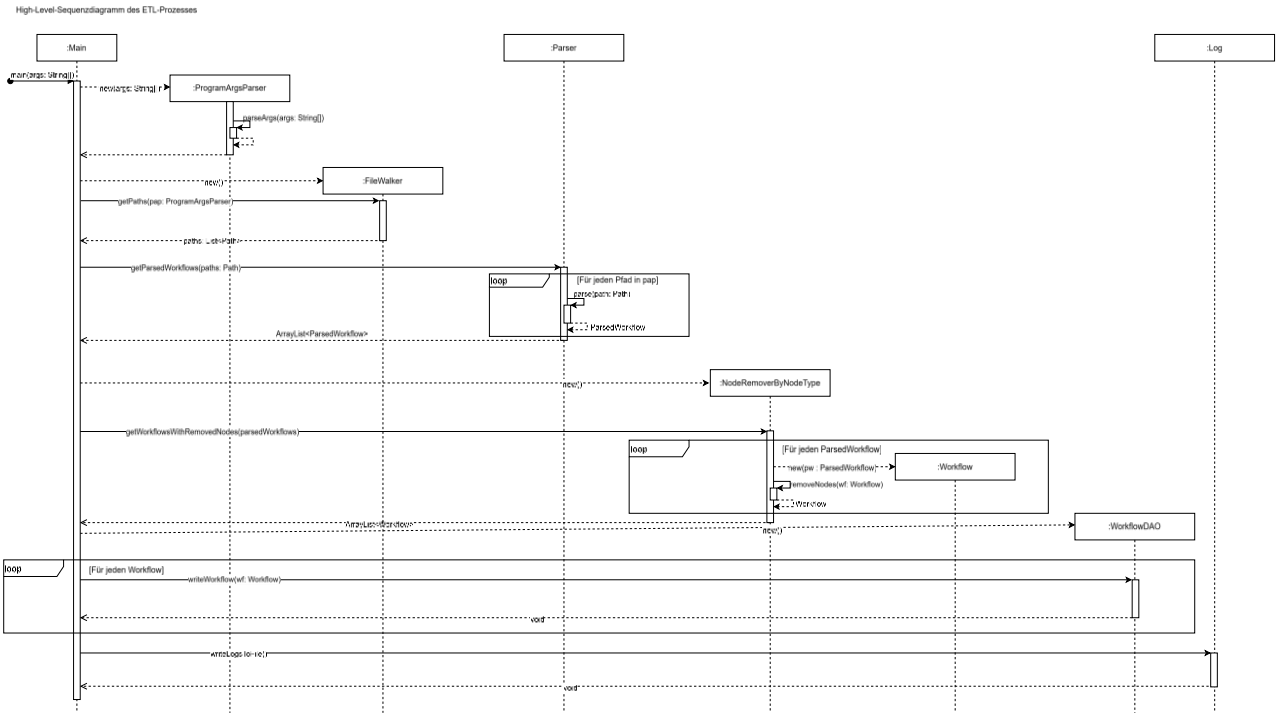
Phase: Dokumentation (8h)

Dokumentation erstellen	8h

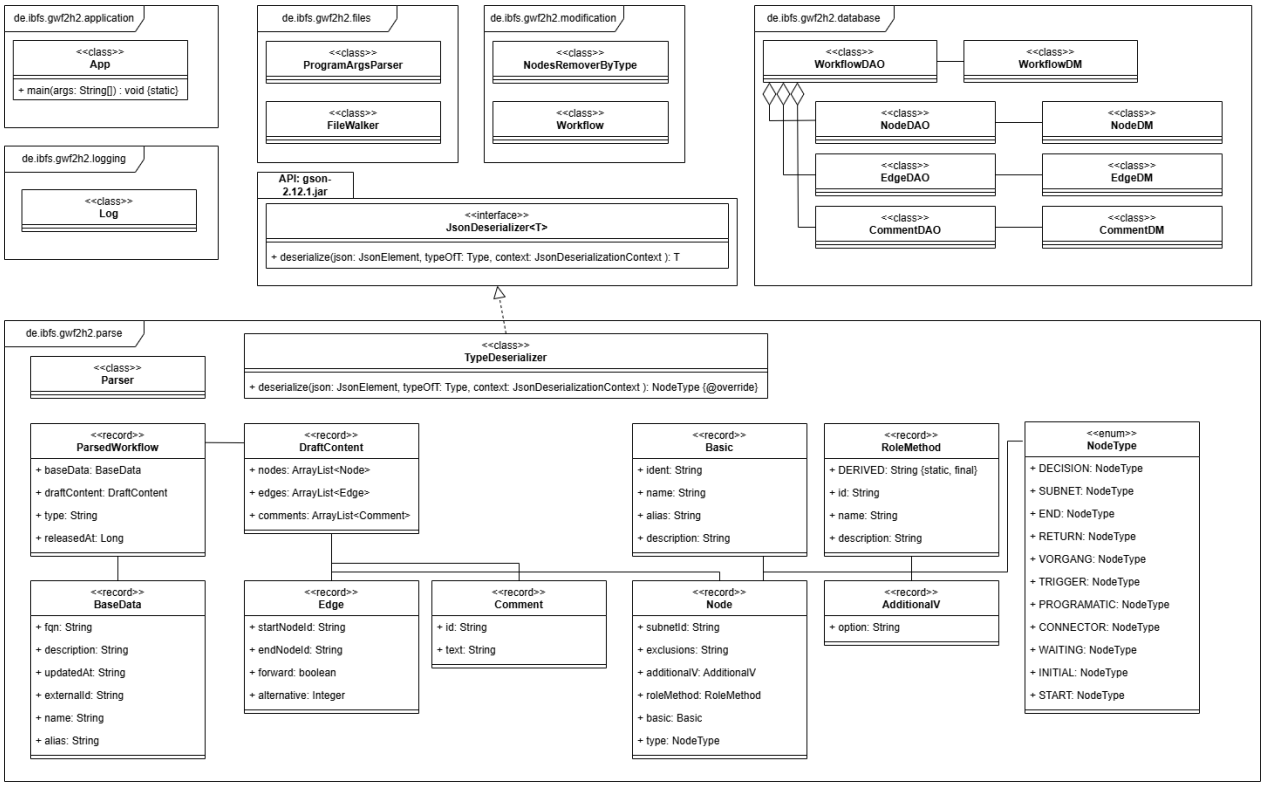
5.2 Use-Case-Diagramm



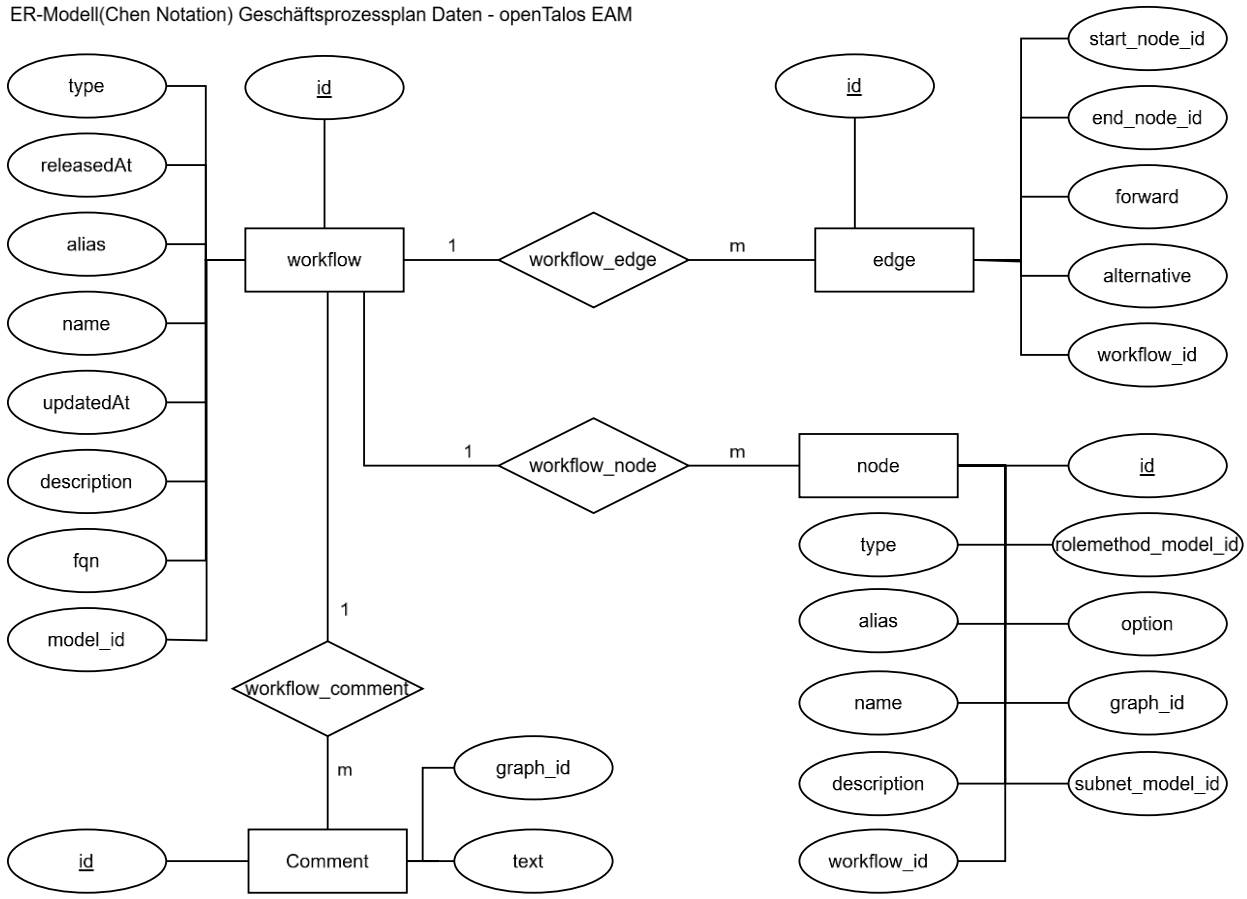
5.3 High-Level-Sequenzdiagramm des ETL-Prozesses



5.4 Klassendiagramm ETL-Programm

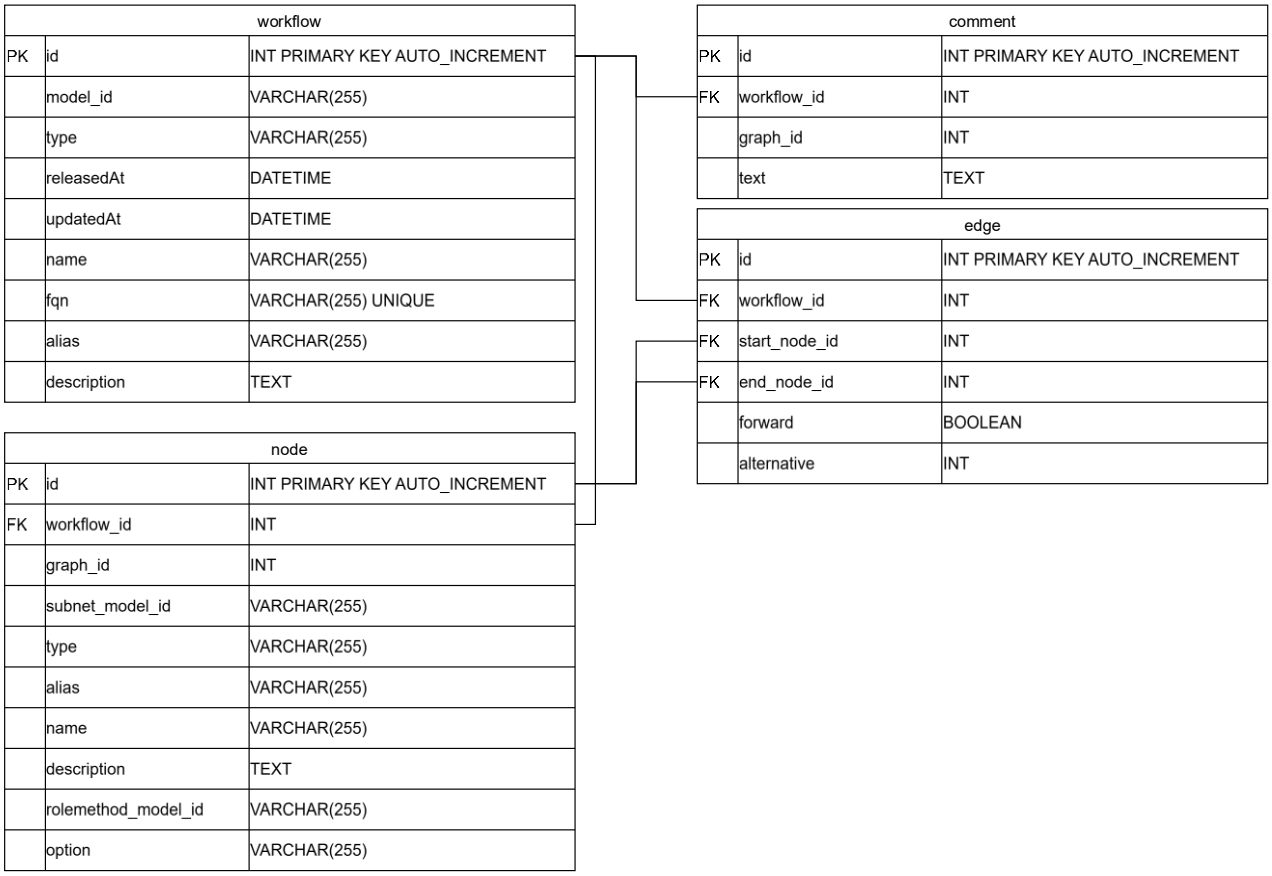


5.5 ER-Modell


ER-Modell(Chen Notation) Geschäftsprozessplan Daten - openTalos EAM
type
releasedAt
alias
name
updatedAt
description
fqn
model_id
id
workflow
1
workflow_edge
m
id
edge
start_node_id
end_node_id
forward
alternative
workflow_id
1
workflow_node
m
node
id
type
rolemethod_model_id
alias
option
name
graph_id
description
subnet_model_id
workflow_id
1
workflow_comment
m
graph_id
id
Comment
text

5.6 Datenbankschema

Datenbankschema Geschäftsprozessplandaten - openTalos EAM

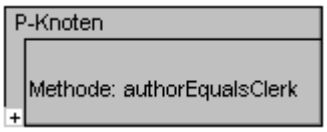
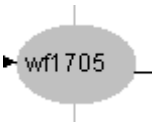
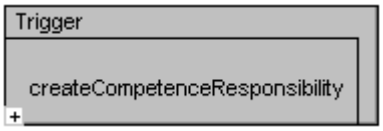
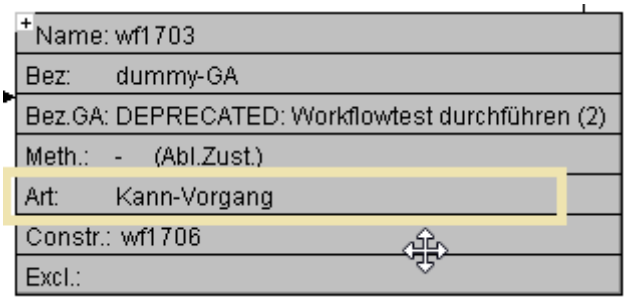


5.7 Screenshot H2 Konsole mit Auszug der eingelesenen Daten

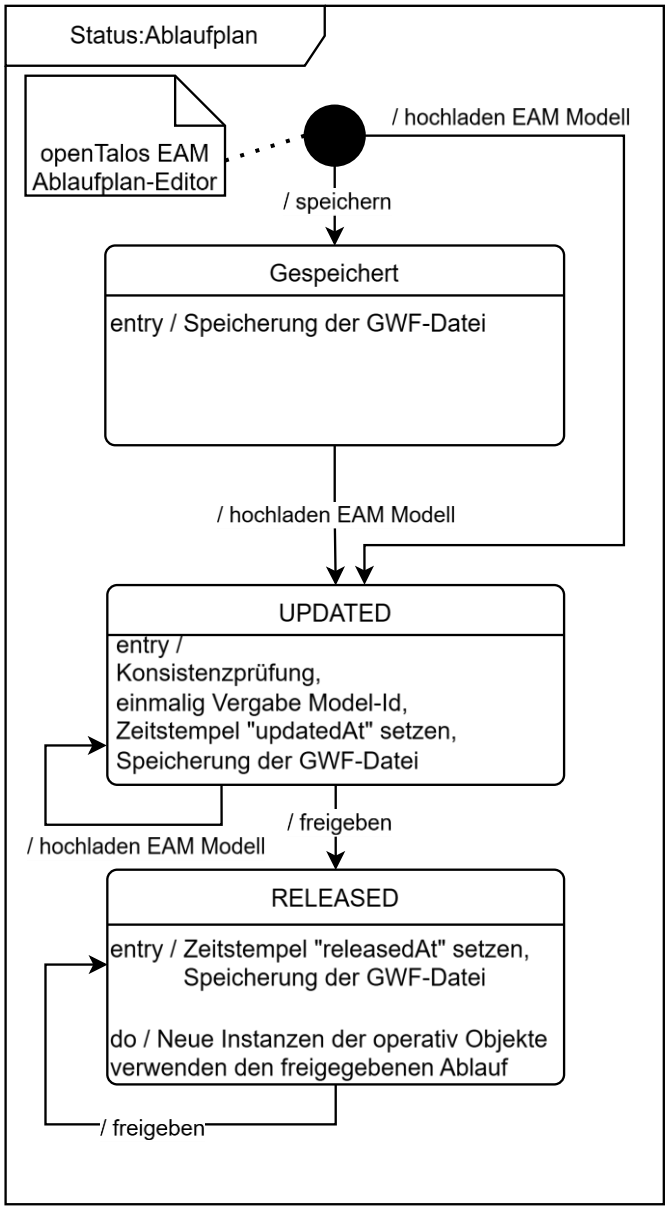
123	500:506[redacted]:WF	MAIN	wf10	wfadmin.wf10
124	500:506[redacted]:WF	SUB	wf11	wfadmin.wf11
125	500:671[redacted]:WF	SUB	wf12	wfadmin.wf12
126	500:901[redacted]:WF	SUB	wf13	wfadmin.wf13
127	500:160[redacted]:WF	SUB	wf14	wfadmin.wf14
128	500:160[redacted]:WF	SUB	wf15	wfadmin.wf15
129	500:172[redacted]:WF	SUB	wf16	wfadmin.wf16
130	500:172[redacted]:WF	SUB	wf17	wfadmin.wf17
131	661:3689[redacted]:WF	MAIN	WorkflowMessage_01	wfadmin.WorkflowMessage_01
132	500:161[redacted]:WF	SUB	NormalizationStepImpl_[redacted]_tn01	workclearance.[redacted]NormalizationStepImpl_[redacted]
133	500:161[redacted]:WF	MAIN	NormalizationStepImpl_[redacted]	workclearance.[redacted].NormalizationStepImpl_hn
134	500:1187[redacted]:WF	SUB	WorkClearanceImpl[redacted]n01	workclearance.[redacted]WorkClearanceImpl_[redacted]
135	500:1716[redacted]:WF	SUB	WorkClearanceImpl_[redacted]_tn02	workclearance.[redacted].WorkClearanceImpl_[redacted]
136	500:1186[redacted]:WF	MAIN	WorkClearanceImpl_hn	workclearance.[redacted]WorkClearanceImpl_hn
137	500:120[redacted]:WF	SUB	WorkClearancePositionImpl[redacted]	workclearance.[redacted]WorkClearancePositionImpl[redacted]
138	500:120[redacted]:WF	SUB	WorkClearancePositionImpl[redacted]	workclearance.[redacted]WorkClearancePositionImpl[redacted]
139	500:119[redacted]:WF	MAIN	WorkClearancePositionImpl_hn	workclearance.[redacted].WorkClearancePositionImpl_hn
140	500:160[redacted]:WF	SUB	WorkClearanceStepImpl[redacted]tn01	workclearance.[redacted]WorkClearanceStepImpl[redacted]
141	500:160[redacted]:WF	MAIN	WorkClearanceStepImpl_hn	workclearance.[redacted]WorkClearanceStepImpl_hn
142	500:161[redacted]:WF	SUB	NormalizationStepImpl_tn01	workclearance.NormalizationStepImpl_tn01
143	500:1187[redacted]:WF	SUB	WorkClearanceImpl_tn01	workclearance.WorkClearanceImpl_tn01
144	500:1195[redacted]:WF	SUB	WorkClearancePositionImpl_tn01	workclearance.WorkClearancePositionImpl_tn01
145	500:120[redacted]:WF	SUB	WorkClearancePositionImpl_tn02	workclearance.WorkClearancePositionImpl_tn02
146	500:160[redacted]:WF	SUB	WorkClearanceStepImpl_tn01	workclearance.WorkClearanceStepImpl_tn01

(146 Datensätze, 1 ms)

5.8 Auszug Bedienungsanleitung: optionale Bereinigung von technischen Knoten

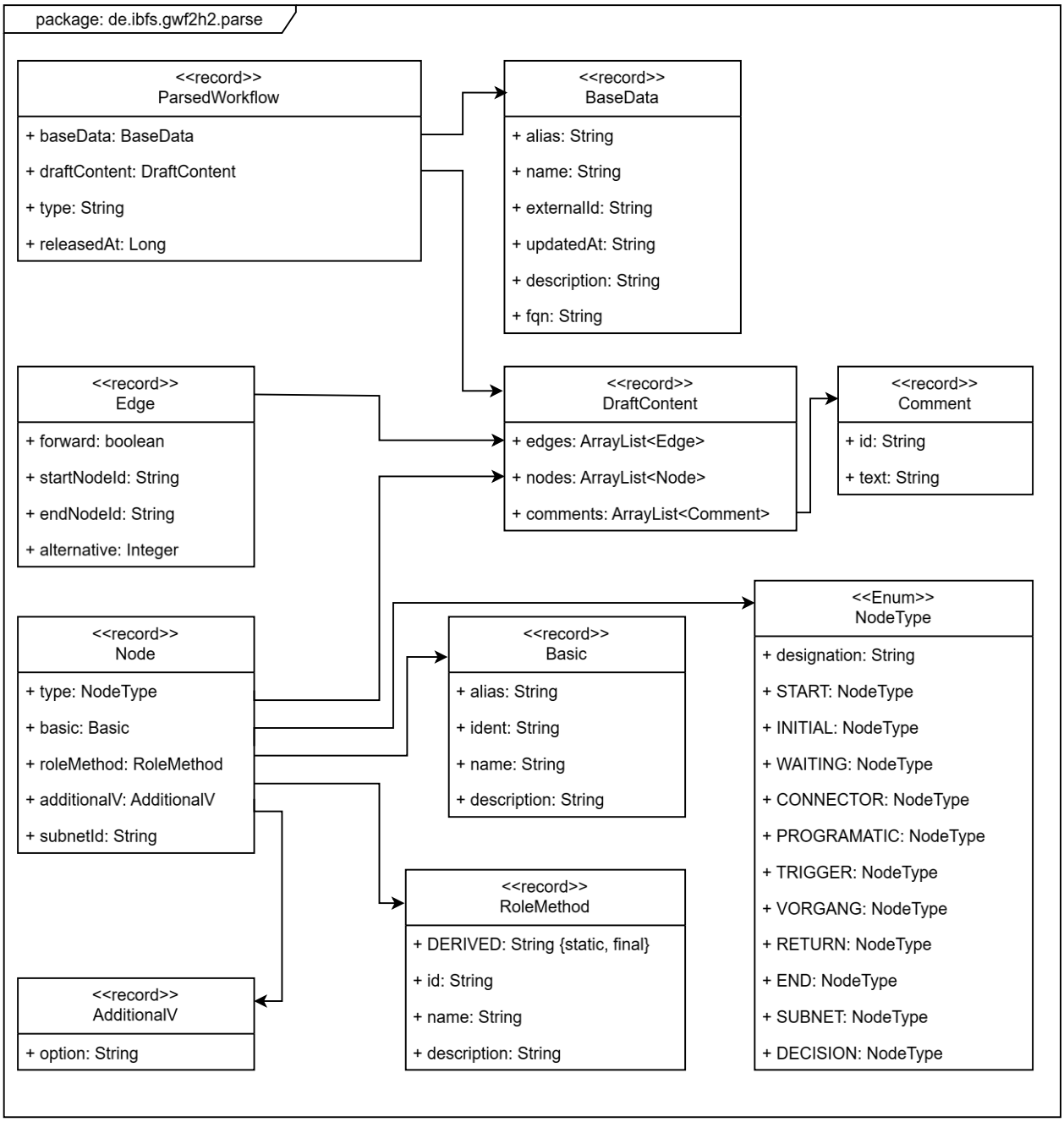
OPEN TALOS Typ	OPEN TALOS Symbol
P-Knoten	
Konnektoren	
Trigger	
Kann-Vorgänge	

5.9 Zustandsdiagramm: Statusverlauf eines Ablaufplans

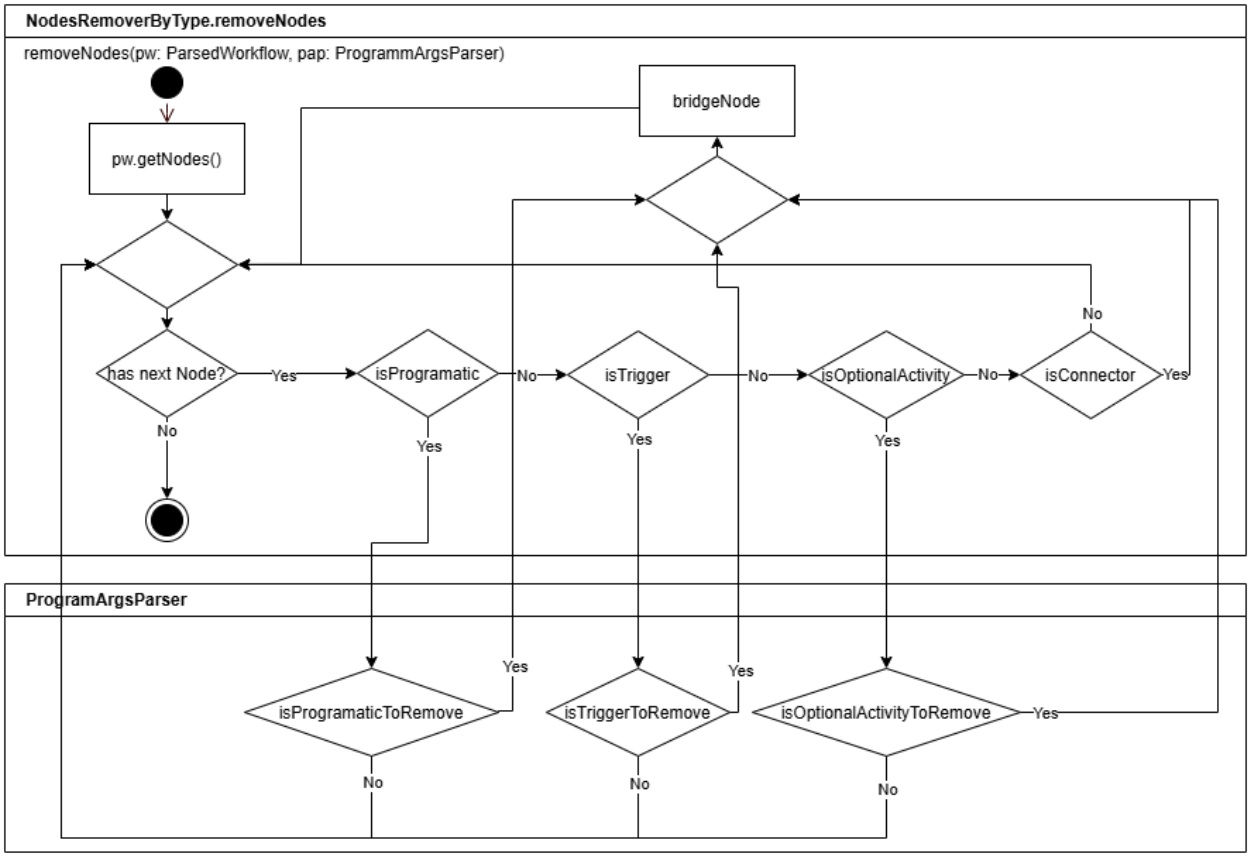


5.10 Klassendiagramm der relevanten Attribute aus der JSON-Hierarchie

Relevante Attribute in der JSON-Hierachie (gwf-Dateien)



5.11 Aktivitätsdiagramm der Graph-Daten Reduzierung



5.12 Auszug GWF-Datei

Der Screenshot zeigt einen Auszug aus einer GWF-Datei (1223 Zeilen) zu **einem**, **kleinen** Ablaufplan.


```
1174            "activity": "500:92050:BA"
1175          }
1176        },
1177        {
1178          "version": 2,
1179          "type": "DECISION",
1180          "basic": {
1181            "version": 1,
1182            "ident": "9",
1183            "name": "wfl704",
1184            "alias": "EP immer false",
1185            "description": ""
1186          },
1187          "caluclated": {
1188            "version": 1,
1189            "refNr": 6,
1190            "initState": 0,
1191            "numberPre": 2,
1192            "derivedConstraints": "",
1193            "derivedExclusions": ""
1194          },
1195          "rectangle": {
1196            "point": {
1197              "x": 500.0,
1198              "y": 289.0
1199            },
1200            "dimension": {
1201              "width": 110.0,
1202              "height": 70.0
1203            }
1204          },
1205          "visibilityFlags": {
1206            "version": 1,
1207            "showAll": false,
1208            "showInfo": false,
1209            "showText": false
1210          },
1211          "typeSpecificMethod": {
1212            "version": 1,
1213            "id": "500:84988:MT"
1214          },
1215          "subnetId": null,
1216          "constraints": null,
1217          "exclusions": null,
1218          "boTypeId": null
1219        }
1220      ]
1221    },
1222    "extensionMode": false
1223  }
```

5.13 Java Code Listing: Klasse: App

```
package de.ibfs.gwf2h2.application;

import java.nio.file.Path;
import java.sql.SQLException;
import java.time.Duration;
import java.time.Instant;
import java.util.List;
import java.util.concurrent.atomic.AtomicInteger;

import de.ibfs.gwf2h2.database.WorkflowDAO;
import de.ibfs.gwf2h2.files.FileWalker;
import de.ibfs.gwf2h2.files.ProgramArgsParser;
import de.ibfs.gwf2h2.logging.Log;
import de.ibfs.gwf2h2.modification.NodesRemoverByType;
import de.ibfs.gwf2h2.modification.Workflow;
import de.ibfs.gwf2h2.parse.ParsedWorkflow;
import de.ibfs.gwf2h2.parse.Parser;

/***
 * Einstiegsklasse der Application
 * @author Sven Kanter, 20.10.2025
 * */
public class App {

    public static void main(String[] args) throws Exception {
        Log.get().register(App.class);
        Instant start = Instant.now();
        Log.get().info(App.class, "Starte ETL-Prozess: Ge-
schäftsprozessplan-Daten Bereitstellung");

        ProgramArgsParser pap = new ProgramArgsParser(args);

        FileWalker fileWalker = new FileWalker();
        List<Path> paths = fileWalker.getPaths(pap);
        Log.get().info(App.class, "Gefundene und Weiterzuverar-
beitende GWF-Dateipfade: " + paths.size());

        List<ParsedWorkflow> parsedWorkflows = Parser.getPar-
sedWorkflows(paths);
```

```
        Log.get().info(App.class, "Erfolgreich geparste Ge-
schäftsprozesspläne: " + parsedWorkflows.size());

        NodesRemoverByType remover = new NodesRemover-
ByType(pap);
        List<Workflow> workflowsWithRemovedNodes = remover.get-
WorkflowsWithRemovedNodes(parsedWorkflows);
        Log.get().info(App.class, "Erfolgreich reduzierte Gra-
phendaten: " + workflowsWithRemovedNodes.size());

        int sumPersistedWorkflows = writeWorkflows(work-
flowsWithRemovedNodes);
        if(sumPersistedWorkflows == paths.size()) {
          Log.get().info(App.class, "Erfolgreich persistierte
Geschäftsprozessplan-Daten: " + sumPersistedWorkflows);
        }
        else {
          Log.get().info(App.class, "Es konnten nicht alle ange-
forderten Geschäftsprozessplan-Daten persistiert werden.\nBitte
die Protokolldatei auf Fehler prüfen" );
        }

        Log.get().info(App.class,"(" + sumPersistedWorkflows + "
/ " + paths.size() + ") Geschäftsprozessplan-Daten wurden er-
folgreich erstellt oder aktualisiert.");
        Log.get().info(App.class, "Dauer der Bereitstellung: ["
+ Duration.between(start, Instant.now()).toMillis() + " ms]");
        Log.get().writeLogsToFile();
        Log.get().info(App.class, "ETL-Prozess abgeschlossen.
Die Protkolldatei des Durchlaufs wurde gespeichert: " +
Log.DEFAULT_LOGFILE_DIRECTORY.toAbsolutePath());
    }

   private static int writeWorkflows(List<Workflow> work-
flowsWithRemovedNodes) throws Exception, SQLException {
          int sumPersistedWorkflows;
          try (WorkflowDAO dao = new WorkflowDAO()) {
            AtomicInteger count = new AtomicInteger(0);
            workflowsWithRemovedNodes.forEach(wf -> {
                try {
                    dao.writeWorkflow(wf);
                    count.incrementAndGet();
                }
```

```
                catch (Exception e) {
                    Log.get().error(App.class, "Fehler beim
Schreiben des Workflows '" + wf + "' in die Datenbank.\n" +
e.getMessage());
                }
            });
            sumPersistedWorkflows = count.intValue();
        }
          return sumPersistedWorkflows;
   }

   @SuppressWarnings("unused")
   private static void deleteAllWorkflows() throws SQLException,
Exception  {
          try (WorkflowDAO dao = new WorkflowDAO()) {
                dao.getAllWorkflowDMs().forEach(wfdm -> {
                      try {
                            dao.deleteWorkflow(wfdm.getId());
                      } catch (SQLException e) {
                            Log.get().error(App.class, "Fehler
beim Löschen aller Daten." + e.getMessage());
                      }
                });
          }

   }

}
```

5.14 Java Code Listing Klasse: ProgramArgsParser

```
package de.ibfs.gwf2h2.files;

import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import java.util.Map.Entry;
import java.util.stream.Collectors;

import de.ibfs.gwf2h2.database.WorkflowDAO;
import de.ibfs.gwf2h2.logging.Log;

/***
 * Die Programmparameter werden interpretiert und die Optionen
bereitgestellt.
 *
 * @author Sven Kanter, 20.10.2025
 */
public class ProgramArgsParser {

    public static final Path DEFAULT_DIRECTORY = Paths.get(Sys-
tem.getProperty("user.dir"));

    public static final String GWF_FILE_EXTENSION = ".gwf";

    public static final String P_EXCLUDE_OPTION = "P";

    public static final String T_EXCLUDE_OPTION = "T";

    public static final String K_EXCLUDE_OPTION = "K";

    public static final String CO_EXCLUDE_OPTION = "CO";

    private static final List<String> IGNORED_FOLDERS_PRODUCTIV
= List.of("\\classes\\", "\\test\\");

    private static final List<String> IGNORED_FOLDERS_TEST =
List.of("\\classes\\");
```

```
    private static final String MODEL_PROJECT_FOLDER_NAME =
"src";

    private Path inputPath;

    private Path outputPath;

    private final List<String> selectedFiles = new Array-
List<>();

    private final Map<String, Boolean> excludeOptions = new
java.util.HashMap<>();
    {
        excludeOptions.put(P_EXCLUDE_OPTION, false);
        excludeOptions.put(T_EXCLUDE_OPTION, false);
        excludeOptions.put(K_EXCLUDE_OPTION, false);
        excludeOptions.put(CO_EXCLUDE_OPTION, true);
    }

    private boolean isTestMode = false;

    /**
     * Die Programmparameter werden interpretiert und die Optio-
nen bereitgestellt.
     *
     * @param args
     */
    public ProgramArgsParser(String[] args) {
        Log.get().register(getClass());
        parseArgs(args);
        Log.get().info(getClass(), toString());
    }

    private void parseArgs(String[] args) {
        if (args == null || args.length == 0) {
            inputPath = DEFAULT_DIRECTORY;
            outputPath = DEFAULT_DIRECTORY;
            return;
        }

        for (int i = 0; i < args.length; i++) {
```

```
            switch (args[i]) {
                case "-e":
                    if (i + 1 < args.length) {
                        inputPath = Paths.get(args[++i]);
                    }
                    break;
                case "-a":
                    if (i + 1 < args.length) {
                        outputPath = Paths.get(args[++i]);
                    }
                    break;
                case "-n":
                    while (i + 1 < args.length && isWorkflow-
FileName(args[i + 1])) {
                        selectedFiles.add(args[++i]);
                    }
                    break;
                case "-exclude":
                    handleExcludeOptions(args, i);
                    break;
                default:
                    if (args[i].endsWith(".gwf")) {
                        Path filePath = Paths.get(args[i]);

                        inputPath = findSrcDirectory(filePath);
                        selectedFiles.add(filePath.getFile-
Name().toString());
                    }
                    break;
            }
        }

        if (inputPath == null) {
            inputPath = DEFAULT_DIRECTORY;
        }

        if (outputPath == null) {
            outputPath = DEFAULT_DIRECTORY;
        }
    }
```

```
    private void handleExcludeOptions(String[] args, int i) {
        for (int j = 0; j < excludeOptions.size(); j++) {
            if (j + i + 1 < args.length && isExcludeOption-
Present(args[j + i + 1])) {
                String excludeOption = args[j + i + 1].toUpper-
Case();
                excludeOptions.put(excludeOption, true);
            }
        }
    }

    private boolean isExcludeOptionPresent(String key) {
        return excludeOptions.containsKey(key.toUpperCase());
    }

    public boolean isExcluded(String key) {
        if (excludeOptions.containsKey(key.toUpperCase())) {
            return excludeOptions.get(key);
        }
        return false;
    }

    private static boolean isWorkflowFileName(String arg) {
        boolean hasDot = arg.contains(".");
        boolean hasWorkflowName = !arg.contains(".") &&
!arg.endsWith(GWF_FILE_EXTENSION) && !arg.contains("-");
        return hasDot || hasWorkflowName;
    }

    public Path getInputPath() {
        return inputPath;
    }

    public Path getOutputPath() {
        return outputPath;
    }

    public List<String> getSelectedFiles() {
        return selectedFiles;
    }
```

```
    public Map<String, Boolean> getExcludeOptions() {
        return excludeOptions;
    }

    private static Path findSrcDirectory(Path filePath) {
        Path current = filePath.toAbsolutePath().getParent();
        while (current != null) {
            if (current.getFileName().toString().equalsIgnore-
Case(MODEL_PROJECT_FOLDER_NAME)) {
                return current;
            }
            current = current.getParent();
        }
        return null;
    }

    private String getSelectedFilesIdentifier() {
        String preInfoMessage = "Selektierte Geschäftsprozess-
pläne: ";
        List<String> files = getSelectedFiles();
        if (files.isEmpty()) {
            return preInfoMessage + "Alle Geschäftsprozesspläne
im Leseverzeichnis\n";
        }
        return preInfoMessage + getSelectedFiles().stream().col-
lect(Collectors.joining(", ")) + "\n";
    }

    private String getExcludedOptionsIdentifier() {
        String excludedOptionsString = "Reduzierte Knoten <-
exclude>: ";
        if (excludeOptions.values().stream().all-
Match(Boolean.FALSE::equals)) {
            return excludedOptionsString + " Keine";
        }
        else {
            String allOptionsString = excludeOptions.entrySet()
                .stream()
                .filter(Entry::getValue)
```

```
                .map(entry -> getExcludeOptionDescrip-
tion(entry.getKey()))
                .collect(Collectors.joining(", "));
            return excludedOptionsString + allOptionsString;
        }
    }

    private static String getExcludeOptionDescription(String
excludeOptionKey) {
        return switch (excludeOptionKey) {
            case P_EXCLUDE_OPTION  -> "P-Knoten";
            case T_EXCLUDE_OPTION  -> "Trigger Knoten";
            case K_EXCLUDE_OPTION  -> "Kann-Vorgang-Knoten";
            case CO_EXCLUDE_OPTION -> "Konnektoren";
            default                -> throw new IllegalArgumen-
tException("Unexpected value: " + excludeOptionKey);
        };
    }

    public boolean isTestMode() {
        return isTestMode;
    }

    public void setTestMode(boolean isTestMode) {
        this.isTestMode = isTestMode;
    }

    public List<String> getIgnoredFolders() {
        return isTestMode ? IGNORED_FOLDERS_TEST : IGNORED_FOL-
DERS_PRODUCTIV;
    }

    @Override
    public String toString() {
        String inputPathString = "\tEingabe-/Leseverzeichnis <-
e>: " + getInputPath().toString() + "\n";
        String dbURL = "\tDB URL: " + WorkflowDAO.DEFAULT_DB_DI-
RECTORY + "\n";
        String ignoredFoldersString = "\tIgnorierte Verzeich-
nisse: " + getIgnoredFolders().stream().collect(Collec-
tors.joining(", "))
                + "\n";
```

```
        String excludeOptionString = "\t" + getExcludedOptionsI-
dentifier() + "\n";
        String selectedFilesString = "\t" + getSelectedFilesI-
dentifier();


        return "\n" + inputPathString + dbURL + ignoredFolders-
String + excludeOptionString + selectedFilesString;
    }


}
```

5.15 Java Code Listing Klasse ProgramArgsParserTest

```
package de.ibfs.gwf2h2.files;

import static org.junit.jupiter.api.Assertions.assertDoesNotT-
hrow;
import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.junit.jupiter.api.Assertions.assertNotNull;
import static org.junit.jupiter.api.Assertions.assertTrue;

import org.junit.jupiter.api.Test;

class ProgramArgsParserTest {

    @Test
    void testProgramArgsParserrExists() {
        String[] args = {};
        ProgramArgsParser pap = new ProgramArgsParser(args);
        assertNotNull(pap);
    }

    @Test
    void testEmptyArgsDoesNotThrow() {
        String[] args = {};
        assertDoesNotThrow(() -> new ProgramArgsParser(args));
    }

    @Test
    void testUseDefaultReadingDirectoryWithNoArgs() {
        String[] args = {};
        ProgramArgsParser pap = new ProgramArgsParser(args);
        assertEquals(ProgramArgsParser.DEFAULT_DIREC-
TORY.toString(), pap.getInputPath().toString());
    }

    @Test
    void testGetReadingDirectoryFromOption() {
        String tempTestPath = "D:\\";
        String[] args = {"-e", tempTestPath};
        ProgramArgsParser pai = new ProgramArgsParser(args);
```

```
        assertEquals(tempTestPath, pai.getInput-
Path().toString());
    }

    @Test
    void testGetDatabaseDirectoryFromOption() {
        String tempTestPath = "D:\\";
        String[] args = {"-a", tempTestPath};
        ProgramArgsParser pai = new ProgramArgsParser(args);
        assertEquals(tempTestPath, pai.getOutput-
Path().toString());
    }

    @Test
    void testGetSelectedFile() {
        String tempTestFilename = "do.message.Test";
        String[] args = {"-n", tempTestFilename};
        ProgramArgsParser pai = new ProgramArgsParser(args);
        assertEquals(tempTestFilename, pai.getSelectedFi-
les().get(0));
    }

    @Test
    void testGetSelectedFileWithExtension() {
        String tempTestFilename = "do.message.Test.gwf";
        String[] args = {"-n", tempTestFilename};
        ProgramArgsParser pai = new ProgramArgsParser(args);
        assertEquals(tempTestFilename, pai.getSelectedFi-
les().get(0));
    }

    @Test
    void testGetSelectedFiles() {
        String tempTestFilename = "do.message.Test";
        String tempTestFilename2 = "do.info.Test";
        String[] args = {"-n", tempTestFilename, tempTestFile-
name2};
        ProgramArgsParser pai = new ProgramArgsParser(args);
        assertEquals(tempTestFilename, pai.getSelectedFi-
les().get(0));
```

```
        assertEquals(tempTestFilename2, pai.getSelectedFi-
les().get(1));
    }

    @Test
    void testGetSelectedFilesWithExtensions() {
        String tempTestFilename = "do.message.Test.gwf";
        String tempTestFilename2 = "do.info.Test.gwf";
        String[] args = {"-n", tempTestFilename, tempTestFile-
name2};
        ProgramArgsParser pai = new ProgramArgsParser(args);
        assertEquals(tempTestFilename, pai.getSelectedFi-
les().get(0));
        assertEquals(tempTestFilename2, pai.getSelectedFi-
les().get(1));
    }

    @Test
    void testSetExludeProgramaticNode() {
        String excludeProgrammaticOption = "p";
        String[] args = {"-exclude", excludeProgrammaticOption};
        ProgramArgsParser pai = new ProgramArgsParser(args);
        assertTrue(pai.getExcludeOptions().get("P"));
    }

    @Test
    void testSetExludeTriggerNode() {
        String excludeTriggerOption = "t";
        String[] args = {"-exclude", excludeTriggerOption};
        ProgramArgsParser pai = new ProgramArgsParser(args);
        assertTrue(pai.getExcludeOptions().get("T"));
    }

    @Test
    void testSetExludeOptionalActivity() {
        String excludeOptionalActivity = "k";
        String[] args = {"-exclude", excludeOptionalActivity};
        ProgramArgsParser pai = new ProgramArgsParser(args);
        assertTrue(pai.getExcludeOptions().get("K"));
    }
```

```
    @Test
    void testSetMixedExcludeOptions() {
        String excludeProgrammaticOption = "p";
        String excludeTriggerOption = "t";
        String excludeOptionalActivity = "k";
        String[] args = {"-exclude", excludeOptionalActivity,
excludeTriggerOption, excludeProgrammaticOption};
        ProgramArgsParser pai = new ProgramArgsParser(args);
        assertTrue(pai.getExcludeOptions().get("K"));
        assertTrue(pai.getExcludeOptions().get("T"));
        assertTrue(pai.getExcludeOptions().get("P"));
    }

    @Test
    void testSetAllArgsMixed() {
        String tempTestPath = "D:\\";

        String tempTestFilename = "do.message.Test.gwf";
        String tempTestFilename2 = "do.info.Test";

        String excludeProgrammaticOption = "p";
        String excludeTriggerOption = "t";
        String excludeOptionalActivity = "k";
        String[] args = {"-a", tempTestPath, "-exclude",
excludeOptionalActivity, excludeTriggerOption, excludeProgramma-
ticOption, "-n",
                tempTestFilename, tempTestFilename2, "-e", temp-
TestPath};

        ProgramArgsParser pai = new ProgramArgsParser(args);

        assertEquals(tempTestPath, pai.getInput-
Path().toString());
        assertEquals(tempTestPath, pai.getOutput-
Path().toString());

        assertEquals(tempTestFilename, pai.getSelectedFi-
les().get(0));
        assertEquals(tempTestFilename2, pai.getSelectedFi-
les().get(1));

        assertTrue(pai.getExcludeOptions().get("K"));
```

```
        assertTrue(pai.getExcludeOptions().get("T"));
        assertTrue(pai.getExcludeOptions().get("P"));
    }

    private static ProgramArgsParser getWithAllArgs() {
        String tempTestPath = "D:\\";

        String tempTestFilename = "do.message.Test.gwf";
        String tempTestFilename2 = "do.info.Test";

        String excludeProgrammaticOption = "p";
        String excludeTriggerOption = "t";
        String excludeOptionalActivity = "k";
        String[] args = {"-a", tempTestPath, "-exclude",
excludeOptionalActivity, excludeTriggerOption, excludeProgramma-
ticOption, "-n",
                tempTestFilename, tempTestFilename2, "-e", temp-
TestPath};

        return new ProgramArgsParser(args);
    }

    @Test
    void testPrintUsedProgrammArgs() {
        ProgramArgsParser pai = getWithAllArgs();
        assertDoesNotThrow(() -> Sys-
tem.out.println(pai.toString()));
    }

}
```

5.16 Java-Code-Listing: Klasse: WorkflowDAO

```
package de.ibfs.gwf2h2.database;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.Timestamp;
import java.util.ArrayList;
import java.util.List;
import java.util.Optional;

import de.ibfs.gwf2h2.logging.Log;
import de.ibfs.gwf2h2.modification.Workflow;
import de.ibfs.gwf2h2.parse.Comment;
import de.ibfs.gwf2h2.parse.Edge;
import de.ibfs.gwf2h2.parse.Node;
import de.ibfs.gwf2h2.parse.ParsedWorkflow;

/***
 *
 * DatabaseAccessObject eines Workflows zur db
 *
 * @author Sven Kanter, 22.10.2025
 */
public class WorkflowDAO implements AutoCloseable {

    private static final String SQL_CREATE_TABLE_WORK-
FLOW_IF_NOT_EXISTS = """
            CREATE TABLE IF NOT EXISTS workflow (
            id INT AUTO_INCREMENT PRIMARY KEY,
            model_id VARCHAR(255),
            type VARCHAR(255),
            name VARCHAR(255),
            fqn VARCHAR(255) UNIQUE,
            alias VARCHAR(255),
            description TEXT,
            releasedAt TIMESTAMP,
```

```
            updatedAt TIMESTAMP
            )""";

    public static final String DEFAULT_DB_DIRECTORY =
"jdbc:h2:./dbh2";

    public static final String ADMIN_USER = "sa";

    private static final String ADMIN_PASS = "";

    private Connection connection;

    /***
     * Standard Konstruktor
     *
     * @throws SQLException
     */
    public WorkflowDAO() throws SQLException {
          Log.get().register(getClass());
        try {
            connection = DriverManager.getConnec-
tion(DEFAULT_DB_DIRECTORY, ADMIN_USER, ADMIN_PASS);
        }
        catch (org.h2.jdbc.JdbcSQLNonTransientConnectionExcep-
tion e) {
          String errorMessage = "Datenbank wird genutzt. Bitte
alle Verbindungen (Web Konsole) trennen und erneut versuchen.\n"
+ e.getMessage();
            Log.get().error(getClass(), errorMessage);
        }
    }

    /***
     * Konstruktor für Testklassen
     *
     * @param connection
     */
    public WorkflowDAO(Connection connection) {
        this.connection = connection;
    }
```

```
    public void writeWorkflow(Workflow wf) throws SQLException {
        createTableIfNotExists();
        if (isWorkflowPresent(wf)) {
            updateWorkflow(getWorkflowDM(wf));
        }
        else {
            createWorkflow(getWorkflowDM(wf));
        }

        Optional<WorkflowDM> workflowDM = selectAndGetWork-
flowDM(wf.getParsedWorkflow().getFullQualifiedName());
        if (workflowDM.isPresent()) {
            NodeDAO nodeDAO = new NodeDAO(connection);
            wf.getNodes().forEach(n -> writeNode(nodeDAO, work-
flowDM.get().getId(), n));

            EdgeDAO edgeDAO = new EdgeDAO(connection);
            wf.getEdges().forEach(e -> writeEdge(edgeDAO, node-
DAO, workflowDM.get().getId(), e));

            CommentDAO commentDAO = new CommentDAO(connection);
            wf.getComments().forEach(c -> writeComment(comment-
DAO, workflowDM.get().getId(), c));
        }

    }

    public void selectAndShowAllNodes() throws SQLException {
        NodeDAO nodeDAO = new NodeDAO(connection);
        List<NodeDM> allNodeDMs = nodeDAO.getAllNodeDMs();
        allNodeDMs.stream().map(NodeDM::toString).forEach(Sys-
tem.out::println);

    }

    private static void writeComment(CommentDAO commentDAO, int
workflowPrimaryKeyId, Comment comment) {
        try {
            commentDAO.writeComment(workflowPrimaryKeyId, com-
ment);
        }
```

```
        catch (SQLException e) {
            System.out.println(e.getMessage());
        }
    }

    private static void writeEdge(EdgeDAO edgeDAO, NodeDAO node-
DAO, int workflowPrimaryKeyId, Edge edge) {
        try {
            NodeDM startNode = nodeDAO.getNodeDM(Integer.par-
seInt(edge.startNodeId()));
            NodeDM endNode = nodeDAO.getNodeDM(Integer.par-
seInt(edge.endNodeId()));

            edgeDAO.writeEdge(workflowPrimaryKeyId, start-
Node.getId(), endNode.getId(), edge);
        }
        catch (SQLException e) {
            System.out.println(e.getMessage());
        }
    }

    private static void writeNode(NodeDAO nodeDAO, int work-
flowPrimaryKeyId, Node node) {
        try {
            nodeDAO.writeNode(workflowPrimaryKeyId, node);
        }
        catch (SQLException e) {
            System.out.println(e.getMessage());
        }
    }

    private boolean createTableIfNotExists() throws SQLException
{
        try (PreparedStatement createTableStatement = connec-
tion.prepareStatement(SQL_CREATE_TABLE_WORKFLOW_IF_NOT_EXISTS))
{
            return createTableStatement.execute();
        }
    }

    private boolean isWorkflowPresent(Workflow wf) throws
SQLException {
```

```
        return getAllWorkflowDMs().stream().anyMatch(wfdm ->
wfdm.getFqn().equals(wf.getParsedWorkflow().getFullQualified-
Name()));
    }

    private static WorkflowDM getWorkflowDM(Workflow wf) {
        ParsedWorkflow pwf = wf.getParsedWorkflow();
        return new WorkflowDM(0, pwf.getModelId(),
pwf.getType(), pwf.getReleasedAt(), pwf.getUpdatedAt(), pwf.get-
Name(),
                pwf.getFullQualifiedName(), pwf.getAlias(),
pwf.getDescription());
    }

    private void createWorkflow(WorkflowDM wfdm) throws
SQLException {
        String sql = "INSERT INTO workflow (model_id, type,
name, fqn, alias, description, releasedAt, updatedAt) VALUES (?,
?, ?, ?, ?, ?, ?, ?)";
        try (PreparedStatement stmt = connection.prepareState-
ment(sql)) {
            stmt.setString(1, wfdm.getModeId());
            stmt.setString(2, wfdm.getType());
            stmt.setString(3, wfdm.getName());

            stmt.setString(4, wfdm.getFqn());
            stmt.setString(5, wfdm.getAlias());
            stmt.setString(6, wfdm.getDescription());
            stmt.setTimestamp(7, new Timestamp(wfdm.getRelease-
dAt()));
            stmt.setTimestamp(8, new Timestamp(wfdm.getUpdate-
dAt()));
            stmt.executeUpdate();
        }
    }

    public Optional<WorkflowDM> selectAndGetWorkflowDM(String
fullQualifiedName) throws SQLException {
        String sql = "SELECT * FROM workflow WHERE fqn = ?";
        List<WorkflowDM> workflows = new ArrayList<>();
        try (PreparedStatement stmt = connection.prepareState-
ment(sql)) {
```

```
            stmt.setString(1, fullQualifiedName);
            try (ResultSet rs = stmt.executeQuery()) {
                while (rs.next()) {
                    workflows.add(new WorkflowDM(rs.ge-
tInt("id"), rs.getString("model_id"), rs.getString("type"),
                            rs.getTimestamp("releasedAt").get-
Time(), rs.getTimestamp("updatedAt").getTime(), rs.get-
String("name"),
                            rs.getString("fqn"), rs.get-
String("alias"), rs.getString("description")));
                }
            }
        }

        if (workflows.size() > 1) {
            throw new UnsupportedOperationException("Inkonsis-
tenz: Workflows mit gleichem voll qualifiziertem Namen in der
Datenbank: " + fullQualifiedName);
        }

        return Optional.of(workflows.get(0));
    }

    public List<WorkflowDM> getAllWorkflowDMs() throws SQLExcep-
tion {
        String sql = "SELECT * FROM workflow";
        List<WorkflowDM> workflows = new ArrayList<>();
        try (PreparedStatement stmt = connection.prepareState-
ment(sql); ResultSet rs = stmt.executeQuery()) {
            while (rs.next()) {
                workflows.add(new WorkflowDM(rs.getInt("id"),
rs.getString("model_id"), rs.getString("type"),
                        rs.getTimestamp("releasedAt").getTime(),
rs.getTimestamp("updatedAt").getTime(), rs.getString("name"),
                        rs.getString("fqn"), rs.get-
String("alias"), rs.getString("description")));
            }
        }
        return workflows;
    }
```

```
    public void updateWorkflow(WorkflowDM workflowDM) throws
SQLException {
        String sql = "UPDATE workflow SET model_id = ?, type =
?, name = ?, fqn = ?, alias = ?, description = ?, releasedAt =
?, updatedAt = ? WHERE id = ?";
        try (PreparedStatement stmt = connection.prepareState-
ment(sql)) {
            stmt.setString(1, workflowDM.getModeId());
            stmt.setString(2, workflowDM.getType());
            stmt.setString(3, workflowDM.getName());
            stmt.setString(4, workflowDM.getFqn());
            stmt.setString(5, workflowDM.getAlias());
            stmt.setString(6, workflowDM.getDescription());
            stmt.setTimestamp(7, new Timestamp(workflowDM.ge-
tReleasedAt()));
            stmt.setTimestamp(8, new Timestamp(workflowDM.ge-
tUpdatedAt()));

            stmt.setInt(9, workflowDM.getId());
            stmt.executeUpdate();
        }
    }

    public void deleteWorkflow(int id) throws SQLException {
          try (Statement stmt = connection.createStatement()) {
            stmt.execute("SET REFERENTIAL_INTEGRITY FALSE");
        }

           NodeDAO nodeDao = new NodeDAO(connection);
           nodeDao.deleteNodes(id);

           EdgeDAO edgeDAO = new EdgeDAO(connection);
           edgeDAO.deleteEdges(id);

           CommentDAO commentDAO = new CommentDAO(connection);
           commentDAO.deleteComments(id);

        String sql = "DELETE FROM workflow WHERE id = ?";
        try (PreparedStatement stmt = connection.prepareState-
ment(sql)) {
            stmt.setInt(1, id);
```

```
            stmt.executeUpdate();
        }

        try (Statement stmt = connection.createStatement()) {
            stmt.execute("SET REFERENTIAL_INTEGRITY TRUE");
        }
    }

    public void truncateAllTables() throws SQLException {
        try (Statement stmt = connection.createStatement()) {
            stmt.execute("SET REFERENTIAL_INTEGRITY FALSE");
        }

        try (PreparedStatement truncateStatement = connec-
tion.prepareStatement("TRUNCATE TABLE workflow")) {
            truncateStatement.executeUpdate();
        }

        NodeDAO nodeDao = new NodeDAO(connection);
        nodeDao.truncateNodeTable();

        EdgeDAO edgeDAO = new EdgeDAO(connection);
        edgeDAO.truncateEdgeTable();

        CommentDAO commentDAO = new CommentDAO(connection);
        commentDAO.truncateCommentTable();

        try (Statement stmt = connection.createStatement()) {
            stmt.execute("SET REFERENTIAL_INTEGRITY TRUE");
        }
    }

    @Override
    public void close() throws Exception {
        if (connection != null && !connection.isClosed()) {
            connection.close();
        }
    }

}
```