IHK Schwaben

Ihre Ansprechpartnerin

Michaela Matz
Fachabteilung Ausbildungsprüfungen
@ michaela.matz@schwaben.ihk.de
0821 3162-1405

Deckblatt für die Projektdokumentation Fachinformatiker/-in – Anwendungsentwicklung

Bitte verwenden Sie dieses Deckblatt immer als erste Seite für Ihre Projektdokumentation.

PROJEKTBEZEICHNUNG:
(bitte ausfüllen)

AUSBILDUNGS-/ UMSCHULUNGSBETRIEB:

Firma
Pflichtfeld

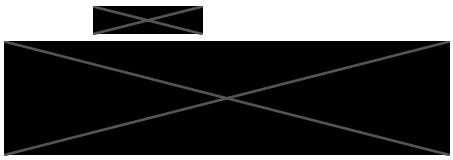
Straße
Pflichtfeld

PLZ, Ort
Pflichtfeld

Projektverantwortlicher
Pflichtfeld

Telefonnummer

E-Mail
Pflichtfeld



PRÜFUNGSTEILNEHMER/-IN:

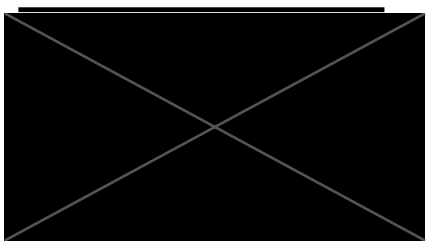
Name, Vorname
Pflichtfeld

Straße
Pflichtfeld

PLZ, Ort
Pflichtfeld

Telefonnummer

E-Mail
Pflichtfeld



PRÜFUNGSNUMMER:
(siehe Genehmigungsschreiben)
Pflichtfeld

PRÜFUNGSJAHR:
(z. B.: Sommer 2022, Winter 2022/23, ...)
Pflichtfeld

IHK Schwaben

Zeitlicher Ablauf

DATUM	TÄTIGKEIT	ZEIT IN STUNDEN

IHK Schwaben

INHALTE DER PROJEKTDOKUMENTATION

- Deckblatt
- Inhaltsverzeichnis (mit Seitennummerierung)
- Thema der Projektarbeit (Projektziel) / Art des Projekts
- Betriebliche Umfeldbeschreibung des Auftragnehmers, -gebers
- Ausganssituation (Auftragsbeschreibung)
- Angabe z. B. der verwendeten Plattform(-en), Software, Anzahl der Endgeräte etc.
- Personal-, Sachmittel-, Termin- und Kostenplanung (Ressourcenplanung)
- Ablaufplanung
- Darstellung der Prozessketten und/- oder Prozess-Schnittstellen
- Umfassende Beschreibung und Begründung der Vorgehensweise, der Entscheidungen, von eventuellen Abweichungen, von Anpassungen und der erzielten Ergebnisse
- Auftragsergebnis (Soll-Ist-Vergleich)
- Übergabe und Abnahme der betrieblichen Projektarbeit
- Notwendige ergänzende Unterlagen, z. B. Protokolle, Gesprächsnotizen, Ablaufpläne, Quellenangaben etc., sind in einem separaten Anhang beizufügen.

Die aufgeführten Punkte müssen sich im Projekt wiederfinden.

WICHTIGE HINWEISE

- Die Projektdokumentation sollte 15 DIN A4-Seiten in üblicher Schriftgröße (z. B. Arial 10 – 12) nicht überschreiten (ohne Anlagen).
- Deckblatt, Impressum, Inhaltsverzeichnis, Abbildungsverzeichnis, Glossar, Quellenverzeichnis, Kundendokumentation und Anlagen zählen nicht zu den 15 Seiten der Projektdokumentation.
- Fremde Quellen einschließlich Recherchen aus dem Internet sind deutlich zu kennzeichnen.
- Alle relevanten Inhalte der betrieblichen Projektarbeit müssen als Inhalt der Projektdokumentation vorhanden sein.
- Zeilenabstand: 1,0 bis maximal 1,5 Zeilen
- Die Dokumentation muss online über die Anwendung **Projektanträge Online** an die IHK Schwaben als eine PDF-Datei (max. 4 MB) übermittelt werden. Für die Übermittlung sind die vor der Antragsstellung erhaltenen Zugangsdaten zu benutzen.
- Werden Auflagen, die der Prüfungsausschuss zum genehmigten Antrag gemacht hat, bei der betrieblichen Projektarbeit nicht erfüllt, führt dies bei der Bewertung generell zu Punktabzug.
- Zu spät eingereichte Unterlagen können zum Nichtbestehen von der Prüfung führen.

BESONDERHEITEN FÜR DEN AUSBILDUNGSBERUF FACHINFORMATIKER/-IN ANWENDUNGSENTWICKLUNG

§§ 12 Prüfungsbereich Planen und Umsetzen eines Softwareprojektes

(1) Im Prüfungsbereich Planen und Umsetzen eines Softwareprojektes besteht die Prüfung aus zwei Teilen.

(2) Im ersten Teil hat der Prüfling nachzuweisen, dass er in der Lage ist,

1. kundenspezifische Anforderungen zu analysieren,
2. eine Projektplanung durchzuführen,
3. eine wirtschaftliche Betrachtung des Projektes vorzunehmen,
4. eine Softwareanwendung zu erstellen oder anzupassen,
5. die erstellte oder angepasste Softwareanwendung zu testen und ihre Einführung vorzubereiten,
6. die Planung und Durchführung des Projektes anforderungsgerecht zu dokumentieren.

Der Prüfling hat eine betriebliche Projektarbeit durchzuführen und mit praxisbezogenen Unterlagen zu dokumentieren. Vor der Durchführung der betrieblichen Projektarbeit hat er dem Prüfungsausschuss eine Projektbeschreibung zur Genehmigung vorzulegen. In der Projektbeschreibung hat er die Ausgangssituation und das Projektziel zu beschreiben und eine Zeitplanung aufzustellen. Die Prüfungszeit beträgt für die betriebliche Projektarbeit und für die Dokumentation mit praxisbezogenen Unterlagen höchstens 80 Stunden.

(3) Im zweiten Teil hat der Prüfling nachzuweisen, dass er in der Lage ist,

1. die Arbeitsergebnisse adressatengerecht zu präsentieren und
2. seine Vorgehensweise bei der Durchführung der betrieblichen Projektarbeit zu begründen.

Der Prüfling hat die betriebliche Projektarbeit zu präsentieren. Nach der Präsentation wird mit ihm ein Fachgespräch über die betriebliche Projektarbeit und die präsentierten Arbeitsergebnisse geführt. Die Prüfungszeit beträgt insgesamt höchstens 30 Minuten. Die Präsentation soll höchstens 15 Minuten dauern.

(4) Bei der Ermittlung des Ergebnisses für den Prüfungsbereich sind die Bewertungen wie folgt zu gewichten:

1. die Bewertung für den ersten Teil mit 50 Prozent und
2. die Bewertung für den zweiten Teil mit 50 Prozent.

IHK Schwaben

Persönliche Erklärung zur betrieblichen Projektarbeit

Ich,

PRÜFUNGSTEILNEHMER/-IN: Pascal Masny

versichere durch meine Unterschrift, dass ich den betrieblichen Auftrag in der laut Ausbildungsverordnung vorgeschriebenen Zeit tatsächlich selbst ausgeführt habe und der Urheber der Dokumentation bin. Alle Textpassagen der Dokumentation wurden von mir selbst formuliert. Textpassagen, die nicht von mir formuliert wurden, sondern Auszüge aus anderen Texten sind, habe ich mit Fußnote gekennzeichnet und die entsprechende Quelle im Quellenverzeichnis angeführt. Eventuelle Zeitunter-/oder -überschreitungen werde ich im Fachgespräch erläutern und begründen.

Mir ist bewusst, dass es sich bei einer nachweisbaren Zuwiderhandlung gegen die o. a. Erklärung um eine Täuschung gem. § 22 Abs. 3 der Prüfungsordnung der IHK Schwaben handelt und meine Dokumentation mit „ungenügend" bewertet wird.

19.05.2023
Ort, Datum

[redacted]
Unterschrift Prüfungsteilnehmer/-in

Wir versichern, die Richtigkeit der o. a. Angaben des/der Prüfungsteilnehmers/-in. Die o. a. Dokumentation stimmt mit dem betrieblichen Auftrag überein.

19.05.2023
Ort, Datum

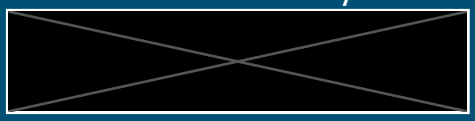
[redacted]
Unterschrift Projektverantworliche/-r

manroland | GOSS

Projektdokumentation

Erstellung einer Python Analyse und Visualisierung von Druckmaschinen Strukturstücklisten und deren Parameter

Fachinformatiker für Anwendungsentwicklung

Auszubildender
Pascal Hendrik Masny


Ausbildungsbetrieb
manroland Goss web systems GmbH
Alois-Senefelder-Allee 1
86153 Augsburg

Prüfnummer
03105

Abgabetermin
21.05.2023

Inhaltsverzeichnis

Abkürzungsverzeichnis

MGWS	manroland Goss web systems GmbH
OSE	Operations Supply Chain Einkauf
pip3	Package Index Python
EMS	Execution Management System
DB	Datenbank
API	Application Programming Interface
AWS	Amazon Web Services
SQL	Structured Query Language
PQL	Process Query Language
HTTP	Hypertext Transfer Protocol
CSV	Comma Separated Values
NaN	Not a Number
MATNR	Materialnummer

Tabellenverzeichnis

1 Projektumfeld

1.1 Betrieb

Seit dem 19. Jahrhundert werden im Zentrum Augsburgs von dem Traditionsunternehmen manroland Goss web systems GmbH (MGWS) Rollen-Offsetdruckmaschinen hergestellt. Die qualitativ hochwertigen Druckerzeugnisse in den Bereichen Zeitungs-, Illustrations- und neuerdings Verpackungsdruck wissen Druckereien auf der ganzen Welt zu schätzen (siehe Abbildung B: VARIOMAN). 2018 fusionierte der Augsburger Standort „manroland web systems GmbH“ mit der amerikanischen Firma „Goss International“, um so eine bessere Marktposition in der Druckindustrie zu erhalten.

1.2 Abteilung

Die Abteilung OSE (Operations Supply Chain Einkauf) ist für den strategischen Einkauf innerhalb der manroland Goss web systems GmbH verantwortlich. Sie stellt sicher, dass alle benötigten Materialien, Komponenten und Dienstleistungen in der erforderlichen Qualität, Menge und zum richtigen Zeitpunkt zur Verfügung stehen. Dabei legt die OSE großen Wert auf die Entwicklung von langfristigen und stabilen Partnerschaften mit Lieferanten, um die besten Konditionen und die kontinuierliche Verbesserung der Lieferketten zu gewährleisten.

1.3 Ausgangssituation

Eine Druckmaschine kann je nach Ausführung aus 20.000 bis 50.000 Baugruppen und 90.000 bis 190.000 einzelnen Bauteilen bestehen. Um den Überblick über die einzelnen Bauteile einer Druckmaschine und deren Preisentwicklung zu erhalten, müssen aufwendige Datensammlungen durchgeführt werden. Diese Datensammlungen sind jedoch ohne IT-Hilfe kaum realisierbar und erfordern viele aufwändige Schritte. Doch auch wenn die Daten erhoben wurden, fehlt oft die Interpretationskraft, um sinnvolle Schlüsse daraus zu ziehen. Zudem werden die Daten oft nicht adäquat visualisiert, was es schwierig macht, eine direkte Verknüpfung zwischen den Daten, der Bauteil-Struktur und deren Parametern herzustellen. Um diese Herausforderungen zu bewältigen, wurde beschlossen, die dafür spezialisierte Software Celonis im Einkauf einzuführen, zur Optimierung der Geschäftsabläufe, um Möglichkeit der digitalen Datenanalyse zu erweitern.

1.4 Durchführungszeitraum

Nach Absprache mit dem Projektverantwortlichen Daniel Schuster, wurde das Projekt zwischen dem 08.05.2023 bis einschließlich dem 19.05.2023 durchgeführt.

1.5 Projektbegründung

Angesichts der in 1.3 Ausgangssituation beschriebenen Herausforderungen im strategischen Einkauf der MGWS wurde dieses Projekt ins Leben gerufen, eine Python-Analyse und Visualisierung von Druckmaschinen Strukturstücklisten und deren Parametern zu entwickeln. Da die strategischen Einkäufer nicht über das erforderliche technische Wissen verfügen, soll die Analyse einen einfachen und schnellen Überblick über die verschiedenen Bauteile und ihrer Parameter wie Preis und Lieferzeit und deren Struktur ermöglichen.

Durch die Automatisierung der Aggregation und Visualisierung der Bauteile und Parameter können wertvolle Zeit und Kosten eingespart werden. Das Ziel des Projekts ist es, den Einkäufern fundierte Entscheidungen auf Basis klarer Daten und Zusammenhänge zu ermöglichen, was letztendlich zur Effizienzsteigerung und Kosteneinsparung im strategischen Einkauf der Firma beitragen wird.

1.6 Technisches Projektumfeld

Bei der MGWS wird eine moderne und leistungsfähige Dateninfrastruktur betrieben, um sämtliche Daten der verschiedenen Geschäftsbereiche effizient zu erfassen, zu verarbeiten und zu analysieren. Hierfür wird die SAP S/4HANA Datenbank eingesetzt, die bei einem dedizierten SAP-Hoster bereitgestellt wird.

Um ausgewählte Daten aus der SAP-Datenbank zu extrahieren und weiterzuverarbeiten, wird ein Java-basierter On-Site-Virtual-Windows-Extractor Server eingesetzt. Dieser Server führt regelmäßig einen stündlichen Cronjob aus, um die ausgewählten Tabellen in einem Cache zu aggregieren. Dadurch werden die Daten schnell und effizient in den nächsten Schritt des Datenflusses überführt.

Im nächsten Schritt werden die Daten vom Celonis-EMS in den Datenpool geladen. Das Celonis-EMS ist eine Softwarelösung zur Analyse von Geschäftsprozessen und wird von der Firma Celonis auf AWS gehostet. Durch die Nutzung des Celonis-EMS können Daten aus verschiedenen Quellen zusammengeführt und analysiert werden, um Prozesse zu optimieren und Einsparpotentiale zu identifizieren.

1.7 Projektschnittstellen

In diesem Projekt besteht eine Herausforderung darin, die Daten aus dem Celonis EMS effektiv abzugreifen, da das System keine direkten HTTP- oder SQL-Schnittstellen zur Verfügung stellt. Um diese Einschränkung zu überwinden und den Zugriff auf die benötigten Daten zu ermöglichen, wird die interne API von Celonis namens PyCelonis verwendet.

PyCelonis ermöglicht es, direkt in Python3 auf die Datapools des Celonis-EMS zuzugreifen und PQL-Abfragen auszuführen. Die Verwendung von PyCelonis in diesem Projekt stellt eine optimale Schnittstelle zwischen der Python3-Programmierung und den Daten im Celonis-EMS sicher. Auf diese Weise können die Daten der Druckmaschinen Strukturstücklisten und deren Parameter effizient abgerufen, verarbeitet und analysiert werden, um die angestrebten Ziele des Projekts zu erreichen.

2 Ist-Analyse

Die Aufgabe besteht darin, eine Softwarelösung zu entwickeln, die den bisher manuellen und komplexen Prozess der Ermittlung von Druckmaschinen Strukturstücklisten und deren Parameter automatisiert und visualisiert.

Es gibt mehrere Herausforderungen, die bewältigt werden müssen. Zum einen erfordert das Finden der relevanten Daten ein tiefgründiges Verständnis der darunterliegenden Datenstruktur. Strategische Einkäufer müssen in der Lage sein, die benötigten Informationen zu lokalisieren, um einzelne Bauteile und deren Parameter identifizieren zu können. Zum anderen gestaltet sich das Zusammenführen der Daten ohne ein hohes Maß an IT-Kenntnissen als schwierig, da die effektive Verarbeitung und Strukturierung der gesammelten Daten davon abhängt.

Des Weiteren verlangt das Verstehen der Zusammensetzung einer Baugruppe aus verschiedenen Baugruppen und Unterteilen umfassendes Wissen im Bereich Mechatronik. Es existiert momentan keine Visualisierung der Daten, was die Analyse und Interpretation der Informationen erschwert. Eine angemessene Visualisierung ist hilfreich, um Muster und Zusammenhänge in den Daten zu erkennen und somit die Entscheidungsfindung im strategischen Einkauf zu unterstützen.

Es gibt eine vorhandene SAP-Transaktion `CSMB`, die für die Ermittlung von Druckmaschinen Strukturstücklisten genutzt wird. Allerdings kann diese nur von geschulten Mitarbeitern mit umfangreichem Wissen über die Baugruppen genutzt werden und erfüllt nur einen Teil der in 1.5 Projektbegründung beschriebenen Anforderungen. Daher ist es wichtig, eine Softwarelösung zu entwickeln, die einfach zu bedienen ist und für alle Mitarbeiter im Fachbereich zugänglich ist. Die neue Softwarelösung sollte auch die vollständige Automatisierung des Prozesses und die Visualisierung der Daten umfassen, um die Effizienz zu steigern und die Entscheidungsfindung zu unterstützen.

3 Projektplanung

3.1 Ressourcenplanung und Recherchen

Das Projekt benötigt verschiedene zusätzliche Bibliotheken und Programme. Die Programmiersprache Python3 ist im Umfeld von „Daten Analyse“ weit verbreitet und bietet diverse Erweiterungen zu diesem Thema, daher wurde diese Programmiersprache ausgewählt. Ein weiterer Vorteil von Python3 ist die Unterstützung von objektorientierter Programmierung (OOP), die es ermöglicht, den Code modularer, wiederverwendbar und einfacher zu warten und zu gestalten. Ein ausschlaggebender Grund für die Wahl von Python3 war zudem die PyCelonis API und das Celonis-EMS, die auf dieser Programmiersprache basieren. Durch diese Integration können wir die Stärken von Python3 in der Datenanalyse optimal nutzen und gleichzeitig auf die spezifischen Funktionen und Fähigkeiten von Celonis zurückgreifen.

So sind in der Tabelle 1: Ressourcen die verwendeten Fremdressourcen aufgelistet. Es ist ein wichtiger Aspekt des Projektes.

Softwarename	Beschreibung
Python 3.10	Compiler für die Programmiersprache Python 3
pip3	Paket installer für Python
anytree	Python-Bibliothek: Arbeiten mit Baumstrukturen
pandas	Python-Bibliothek: Datenmanipulation und -analyse
numpy	Python-Bibliothek: Unterstützung für große, mehrdimensionale Arrays und Matrizen
PyCelonis	Python-Bibliothek: Kommunikation mit dem Celonis-EMS
Graphviz	Software zum Erstellen von Diagrammen und Graphen basierend auf der Graphviz-Sprache DOT
MS Office 2016	Büroanwendungen
VS-Code	Quellcode-Editor für verschiedene Programmiersprachen
Git	Versionskontrollsystem für die Verwaltung von Quellcode-Änderungen
Celonis-EMS	Celonis Execution Management System für Geschäftsprozessanalyse

Tabelle 1: Ressourcen

Während der Recherche für das Projekt stellte sich heraus, dass der vorhandene Firmenlaptop für die Entwicklung des Projekts nicht ausreichend leistungsfähig war. Daher wurden zwei neue Laptops bestellt, um den Anforderungen gerecht zu werden. Dieser zusätzliche Bedarf spiegelt sich in den 6.2 Projektkosten wieder. Die ausgewählten Laptops sind je ein Fujitsu Lifebook E Series mit 32 GB RAM, einem Intel i7-Prozessor mit 10 Kernen, einer 1 TB SSD mit einem Einzelpreis von 1.200,00€.

3.2 Projektphasen Zeitplanung

Die Vorgabe für die Umsetzung des Projektes beträgt 80 Stunden. Vor Projektbeginn wurde das Projekt in Phasen eingeteilt, denen eine geschätzte Dauer aus dem vorgegebenen Kontingent zugewiesen wurde, wie man aus *Tabelle 2: Vorläufige Zeitplanung* entnehmen kann.

Projektphase	Dauer in Stunden
Ist-Analyse	7,5
Projektplanung	12,5
Projektrealisierung	46
Tests und Fehlerbehebungen	4
Projektdokumentation	10

Tabelle 2: Vorläufige Zeitplanung

3.3 Einrichten der Entwicklungsumgebung

Die Entwicklungsumgebung befindet sich auf einem lokalen Windows-10-Pro-PC, welcher den firmeninternen Standards und den Vorgaben entspricht. Die für dieses Projekt benötigten Entwicklungssysteme Visual Studio Code und Python3 wurden über die Installer der offiziellen Organisationswebsite installiert. Diese sind im II. Quellenverzeichnis zu finden. Zudem wurden die in Ressourcenverzeichnis beschriebenen Python3-Bibliotheken und Frameworks über das Python3-Paketverwaltungsprogramm `pip3` installiert und sind global für alle Python-Anwendungen auf diesem PC nutzbar. Dabei wurde darauf geachtet, dass alle installierten Programme und Bibliotheken den firmeninternen Richtlinien und Regeln entsprechen und entsprechend konfiguriert wurden.

Das Versionsverwaltungssystem Git wird verwendet, um den aktuellen Stand des Projekts zu verfolgen. Zur Nutzung muss der Git-Client auf dem Computer installiert werden und das Verzeichnis des Projekts als Git-Repository initialisiert werden. Mit `git add` fügt man Dateien hinzu und `git commit` übernimmt Änderungen im Repository. Eine aussagekräftige Commit-Message ist empfehlenswert. Git ermöglicht eine effektive und systematische Verwaltung von Änderungen an dem Projekt. Als Git-Server wird der firmeninterne Git-Service Gitea genutzt, um eine sichere und zentralisierte Verwaltung des Codes zu gewährleisten.

3.4 Programmgestaltung

Das Programm ist in mehrere Teile gegliedert und umfasst einen Client sowie vier Klassen mit Prozeduren, die für das Aggregieren der Daten, das Filtern der Daten und das Erstellen der Daten-Baumstruktur zuständig sind. Die Baumstruktur ist ein zentrales Element in diesem Projekt, da Druckmaschinen und ihre Baugruppen in einer baumartigen Struktur aufgebaut sind. Eine Baugruppe, wie zum Beispiel ein Druckwerk, besteht aus mehreren Unterbaugruppen, wie Walzen, und jede Walze wiederum besteht aus einzelnen Bauteilen, wie Zylinder, Schrauben usw. Dieses Beispiel ist stark vereinfacht, in der Realität sind die Strukturen weitaus komplexer.

Um die baumartige Datenstruktur effektiv umzusetzen, wird ein spezielles Designpattern benötigt. Hierbei wurde entschieden, das Composite Pattern (siehe dazu: D. Kompositum Entwurfsmuster - (Composite Pattern)) zu verwenden. Dieses Pattern ermöglicht es, eine Hauptklasse (einen Root oder Component), sowie für jede Baugruppe, Unterbaugruppen (Composites) und für jede Unterbaugruppe die einzelnen Bauteile (Leafs) zu haben. Jedes Bauteil referenziert dabei auf die entsprechende (Unter-) Baugruppe und jede Unterbaugruppe auf ihre darüberliegende Baugruppe.

Für die Realisierung der Baumstruktur wird eine spezielle Klasse benötigt. Diese Klasse ist dafür verantwortlich, die Struktur der Baumdaten zu erstellen und die Beziehungen zwischen den verschiedenen Ebenen der Baugruppen, Unterbaugruppen und Bauteile herzustellen. Durch diese Programmgestaltung wird die Komplexität der Druckmaschinen Strukturstücklisten effizient abgebildet und die Verarbeitung der Daten erleichtert.

3.4.1 Aggregation der Daten

Die Aggregation der Daten erfolgt über das Celonis-EMS mithilfe der PyCelonis API. Alle benötigten Tabellen befinden sich im Datenpool "SAP ECC - Inventory Management". Innerhalb dieses Datenpools gibt es ein Datenmodell, das aus mehreren Tabellen und deren Relationen besteht. Für die Aggregation der Daten werden zwei Tabellen benötigt.

Die erste Tabelle, MDSC, ist eine von Celonis vorgegebene Aggregation aus mehreren SAP-Tabellen. Sie enthält die Struktur der Baugruppen, einschließlich der Materialnummer (MATNR) sowie SIGNAL_IN‘s und SIGNAL_OUT‘s, die als Referenzschlüssel für andere Materialien dienen. Die zweite erforderliche Tabelle ist die EKPO-Tabelle, eine SAP-native Tabelle, welche für jede Materialnummer einkaufsspezifische Parameter wie Preise, Lieferzeiten, Konditionen usw. enthält. Jedes Material hat dabei mehrere Einträge, da ein Teil mehrmals bestellt wird.

Während der Aggregation der Daten werden nur bestimmte Felder abgefragt, da nicht alle benötigt werden. Die PyCelonis API verwendet für die Abfrage die Process-Query-Language (PQL), da diese vom Celonis-EMS unterstützt wird. Durch die Verwendung der PyCelonis API und PQL können die relevanten Daten aus den MDSC- und EKPO-Tabellen effizient extrahiert und für die weitere Verarbeitung und Analyse aufbereitet werden.

3.4.2 Filtern der Daten

Da die Datenqualität nicht zu 100% gewährleistet ist, ist es notwendig, die Daten zu filtern. Zunächst sollen alle NaN- und Nullwerte entfernt werden, da diese für die Analyse irrelevant sind. Nachdem die Daten gefiltert wurden, müssen zusätzliche Berechnungen in der EKPO-Tabelle durchgeführt werden, um die erforderlichen Parameter für die Analyse zu erhalten.

Zwei solcher Parameter, die noch nicht in der Tabelle vorhanden sind, sind die prozentuale Veränderung des Preises und die Planlieferzeit. Diese Parameter sind für einen strategischen Einkäufer, insbesondere in dieser wirtschaftlich volatilen Zeit, von großer Bedeutung. Um diese Parameter zu berechnen, wird die Anzahl der letzten `n` Bestellungen herangezogen und dann als Prozentwert hinterlegt.

3.4.3 Erstellung der Datenstruktur

Da die Daten auf dem D. Kompositum Entwurfsmuster - (Composite Pattern) basieren, muss dieses Designmuster in der entsprechenden Klasse implementiert werden. Hierzu soll die Bibliothek "anytree" verwendet werden. Das Composite Pattern ist ein strukturelles Designmuster, das dazu dient, hierarchische, baumartige Strukturen darzustellen, bei denen einzelne Elemente und Gruppen von Elementen auf dieselbe Weise behandelt werden können. Es besteht aus mehreren Komponenten: der Basiskomponente (Root), dem Composite und dem Leaf.

Die Basiskomponente (Root) ist die grundlegende abstrakte Klasse, die sowohl für einzelne Elemente (Leafs) als auch für Gruppen von Elementen (Composites) gemeinsame Methoden und Attribute definiert. Der Composite stellt eine Gruppierung von Elementen dar und erbt von der Basiskomponente. Er verwaltet die Sammlung von Leaf- und Composite-Elementen und ermöglicht es, diese hinzuzufügen, zu entfernen oder zu durchlaufen. Das Leaf repräsentiert einzelne Elemente in der Struktur und erbt ebenfalls von der Basiskomponente.

Die Verwendung des Composite Patterns und der `anytree` Bibliothek ermöglicht die effektive Abbildung der baumartigen Struktur von Druckmaschinen und ihren Baugruppen, wobei jede Baugruppe aus mehreren Unterbaugruppen und diese wiederum aus einzelnen Bauteilen bestehen. Diese Struktur bietet eine klare und leicht verständliche Darstellung der hierarchischen Beziehungen zwischen den verschiedenen Bauteilen und erleichtert die Analyse und Visualisierung der Daten.

3.4.4 Visualisierung der Daten

Nachdem die Datenstruktur erstellt wurde, soll sie zusammen mit all ihren Parametern visualisiert werden. Dafür werden die zugehörigen EKPO-Parameter auf jede Materialnummer angewendet, indem sie miteinander verbunden (gejoined) werden. Die Visualisierung erfolgt anschließend mithilfe von Graphviz, einem Open-Source-Softwarepaket zur Erstellung von Diagrammen basierend auf der Graphzeichensprache DOT. Graphviz ermöglicht die effektive Visualisierung der baumartigen Struktur von Druckmaschinen und ihren Baugruppen sowie der zugehörigen EKPO-Parameter, indem es hierarchische und nicht-hierarchische Graphen und Netzwerkdiagramme automatisch anordnet. Dadurch wird die komplexe Beziehung zwischen den verschiedenen Bauteilen und Parametern anschaulich dargestellt.

4 Projektrealisierung

Die Struktur des Projekts kann der Abbildung C. UML-Klassendiagramm entnommen werden. Zusätzlich kann der Abbildung E. Dateiverzeichnis Struktur, der Aufbau des Codes sowie die I/O-Daten entnommen werden.

4.1 Celonis-EMS

Die Celonis-EMS-Datenbank, auf die zugegriffen wird, ist bereits strukturiert und folgt einer vorgegebenen Organisation. Um auf die Daten zugreifen zu können, wurde ein API-Schlüssel (Celonis API Key) erstellt. Dieser Schlüssel ermöglicht es, Daten abzufragen und ist mit einem technischen Benutzer verknüpft, der nur Leseberechtigungen besitzt. Dies stellt sicher, dass keine unbefugten Änderungen an den Daten vorgenommen werden können.

Recherchen haben ergeben, dass das Celonis-EMS in vielerlei Hinsicht mit einer relationalen Datenbank vergleichbar ist. Daten werden in Tabellen gespeichert, ähnlich wie in SQL-Datenbanken. Besonders hervorzuheben ist, dass die Daten im Celonis EMS in sogenannten Datapools abgelegt werden. Diese Datapools funktionieren ähnlich wie SQL-Datenbanken und dienen als zentraler Speicherort für die strukturierten Daten. Mit der Integration der Datapools und der PQL in das Projekt wird sichergestellt, dass nur die relevanten Daten effizient und präzise abgerufen werden können.

Um auf diese Daten zuzugreifen und sie abzufragen, wird die Abfragesprache PQL verwendet. PQL ist speziell für den Zugriff auf Daten in Celonis-EMS entwickelt worden und ermöglicht eine effiziente und gezielte Abfrage der benötigten Informationen.

4.2 Geschäftslogik Aggregation der Daten

In diesem Abschnitt des Projekts liegt der Fokus auf der Aggregation der Daten aus der Celonis-EMS-Datenbank. Um dies nach Angaben von 3.4.1 Aggregation der Daten zu erreichen, wurde eine separate Klasse namens `CelonisData` (siehe dazu: F. CelonisData.py) erstellt, die für die Verwaltung der Verbindung zu Celonis, das Abrufen von Datenpools und Datenmodellen sowie das Ausführen von Abfragen und Exportieren der Ergebnisse in CSV-Dateien verantwortlich ist.

Der Code des Clients enthält die Initialisierung der `CelonisData`-Klasse und die entsprechenden Anmeldeinformationen wie URL, API-Token und Schlüsseltyp. Durch die Verwendung der `connect_celonis`, `get_data_pool` und `get_data_model` Methoden wird die Verbindung zu Celonis hergestellt und die relevanten Datenressourcen abgerufen.

Anschließend werden PQL-Abfragen erstellt, um Daten aus den gewünschten Tabellen abzurufen. In diesem Beispiel werden zwei Tabellen abgefragt: MDSC und EKPO. Die Abfragen sind so strukturiert, dass sie die erforderlichen Spalten auswählen und die Ergebnisse in zwei separate DataFrames, `result_MDSC` und `result_EKPO`, speichern.

Schließlich werden die `export_to_csv` Methode der `CelonisData`-Klasse verwendet, um die aggregierten Daten in CSV-Dateien zu exportieren, die für die weitere Analyse und Verarbeitung verwendet werden können.

Die Klasse `CelonisData` ermöglicht somit eine modulare und strukturierte Vorgehensweise bei der Datenaggregation und trägt zur Effizienz und Flexibilität des Projekts bei, genau wie in 3.1 Ressourcenplanung und Recherche beschreiben.

4.3 Geschäftslogik Filtern der Daten

Die Datenfilterung wird mithilfe der `DataProcessor`-Klasse (siehe dazu: G. DataProcessor.py) durchgeführt. Der `DataProcessor` ist für das Lesen, Verarbeiten und Zusammenführen der zuvor exportierten CSV-Dateien verantwortlich. Die Klasse verwendet verschiedene Methoden, um den Datensatz für die Analyse vorzubereiten.

Der Code des Clients erstellt eine Instanz der `DataProcessor`-Klasse und gibt den Wert `n` an, der die Anzahl der letzten Bestellungen angibt (wie in 3.4.2 Filtern der Daten beschrieben), die für die Berechnung der Parameter verwendet werden sollen. Dann werden die MDSC- und EKPO-Daten mithilfe der `read_data`-Methode gelesen, mit den `process_mdsc`- und `process_ekpo`-Methoden verarbeitet und anschließend in gefilterten CSV-Dateien gespeichert.

In der `process_mdsc`-Methode werden alle Zeilen mit vorhandenen Werten für `EVENTTIME_T0` entfernt, da diese die Laufzeit einer Baugruppe beschreiben. Wenn eine Baugruppe den Wert `EVENTTIME_T0` beinhaltet, bedeutet dies, dass das Teil eine Neurevision enthält und dieser Datensatz nicht mehr für die Analyse berücksichtigt werden soll. Durch das Filtern dieser Datensätze wird sichergestellt, dass nur die relevanten Daten für die Analyse verwendet werden.

Mit dieser zusätzlichen Anforderung wird die Datenfilterung präziser gestaltet, indem Datensätze mit Neurevisionen ausgeschlossen werden. Dies ermöglichten eine genauere Analyse und die Berücksichtigung der Daten, die für die gewünschten Berechnungen und die daraus resultierenden Erkenntnisse relevant sind. Schließlich wird die `merge_and_save`-Methode verwendet, um die gefilterten MDSC- und EKPO-Daten zusammenzuführen und die Ergebnisse in einer CSV-Datei zu speichern. Die `add_sign`-Methode wird verwendet, um numerischen Werten im Datensatz ein Vorzeichen (+/-) hinzuzufügen.

Die `DataProcessor`-Klasse stellt eine effiziente und strukturierte Vorgehensweise zur Datenfilterung und -bereinigung bereit, die für die anschließende Analyse und Verarbeitung notwendig ist, genau wie in 3.1 Ressourcenplanung und Recherche beschreiben.

4.4 Geschäftslogik Erstellung der Datenstruktur und Visualisierung

Der Code in dieser Phase verwendet das Composite Pattern, um eine Baumstruktur aus den Daten zu erstellen und diese anschließend zu visualisieren. Das Hauptziel besteht darin, eine hierarchische Struktur der Baugruppen (MATNR) basierend auf den Signalen (SIGNAL_IN und SIGNAL_OUT) zu erstellen. Die Klasse `TreeGenerator` (siehe dazu: I. TreeGenerator.py) enthält die Hauptfunktionalität zur Erstellung des Baums, während die Klasse `TreeOperations` (siehe dazu: H. TreeOperations.py) dazu dient, die Baumoperationen auszuführen und die Ergebnisse anzuzeigen. Die `generate_nodes`-Funktion ist verantwortlich für die rekursive Erstellung der Knoten des Baums. Sie nimmt zwei Parameter, `SIGNAL_IN` und ein `parent`-Objekt, um die Kindknoten für das gegebene Signal und den Elternknoten zu identifizieren. Zuerst sucht die Funktion in den Daten nach den entsprechenden Einträgen, bei denen das gegebene Signal einem SIGNAL_OUT entspricht. Wenn solche Einträge gefunden werden, erstellt die Funktion für jeden Eintrag einen neuen Knoten und fügt die entsprechende Zeile zum Ausgabe-DataFrame hinzu.

Anschließend sucht die Funktion nach SIGNAL_IN-Kindknoten für jeden SIGNAL_OUT-Kindknoten. Wenn solche Knoten gefunden werden, fügt sie diese zur Liste der SIGNAL_IN-Kindknoten hinzu und ruft die `generate_nodes`-Funktion rekursiv für jeden SIGNAL_IN-Kindknoten auf. Auf diese Weise wird der Baum rekursiv erstellt, bis keine weiteren Kindknoten gefunden werden.

Die Klasse `TreeOperations` lädt zunächst ein `TreeGenerator`-Objekt und führt dann die verschiedenen Operationen aus, um den Baum zu erstellen und die Ergebnisse anzuzeigen. Der `execute_tree_operations`-Methode werden das `TreeGenerator`-Objekt sowie das Wurzel-SIGNAL_IN und die Wurzel-MATNR übergeben. Die Methode zeigt zunächst den geladenen DataFrame, die Wurzelinformationen und die generierte Baumstruktur an. Anschließend wird der DataFrame des Baums erhalten und angezeigt.

Die `visualize_tree`-Methode in der `TreeGenerator`-Klasse ist verantwortlich für die Visualisierung des generierten Baums und das Speichern der Visualisierung als .dot-Datei, die vom Programm Graphviz visualisiert werden kann. Um dies zu erreichen, verwendet die Methode die DotExporter-Klasse aus der `anytree.exporter`-Bibliothek.

Sie erstellt einen DotExporter-Objekt-Instanz und übergibt den Wurzelknoten des Baums (`self.root`) als Parameter.

Sie ruft die `to_picture-Methode` des `DotExporter`-Objekts auf und übergibt den Pfad und den Dateinamen, unter dem die .dot-Datei gespeichert werden soll. Der Dateiname enthält den Wert von `self.root.name[0] ` (SIGNAL_IN des Wurzelknotens) und die Dateierweiterung .png.

Die `visualize_tree`-Methode speichert die Visualisierung des Baums als .png-Datei im "images" Ordner. Um den Baum mithilfe von Graphviz zu visualisieren, sollte die erzeugte .png-Datei geöffnet werden.

5 Test und Fehlerbehebung

Achtung: Die vorliegenden Testdaten sind firmeninterne Dokumente und unterliegen einer eingeschränkten Nutzung, daher sind sie zensiert. Der Test bezieht sich auf einen White-Box-Ansatz, wobei die Überprüfung ausschließlich auf die Korrektheit der Ergebnisse fokussiert ist. Edge-Cases und Sonderfälle wurden bereits während der 4. Projektrealisierungsphase ausgeschlossen.

5.1 Test mit Testdaten

Zur Datenprüfung wurde die Baugruppe ZSB.FARB-FEUCHTWALZEN WA=1700, B=1300 (Materialnummer 16040003410) ausgewählt (siehe K. ZSB.FARB-FEUCHTWALZEN). Die Materialnummer wurde in die Python-Analyse eingegeben und das Programm ausgeführt. Die dafür angegebenen rohen Testdaten sind in einem Auszug in L. Test-Daten (Auszug) einzusehen. Danach wurden alle zugehörigen Unterbaugruppen und Bauteile ermittelt (M. Programmausgabe 1). Im Anschluss wurden alle erforderlichen Parameter hinzugefügt und visualisiert.
Das Ergebnis der Analyse ist eine baumartige Struktur der Druckmaschinen Strukturstücklisten und deren Parameter, exakt nach den Projektziel und der 2. Ist-Analyse. Das Ergebnis ist O. Visualisierung Ergebnis Raw als .dot Roh-Datei einsehbar, oder auch als Fertige Visualisierung in P. Visualisierung Ergebnis.

5.2 Gegentest SAP

Wie bereits in der 2. Ist-Analyse beschrieben, können Daten nur mit fundiertem Verständnis aus dem SAP-System extrahiert werden. Um das Ergebnis aus zu validieren, wurden die Daten mittels SAP-Transaktion `CSMB` ermittelt. Anschließend wurde das Ergebnis aus CSMB (N. CSMB-Ausgabe) mit dem Ergebnis aus M. Programmausgabe 1, O. Visualisierung Ergebnis Raw sowie P. Visualisierung Ergebnis verglichen. Die Struktur der Bauteile wurde mittels SAP-Transaktion `CSMB` ermittelt, während die Parameter aus den Transaktionen `MD04` und `ZMDISPO` stammen. Die Datenvalidierung bestätigte eine erfolgreiche und korrekte Extraktion.

Es ist wichtig zu beachten, dass in der Darstellung P. Visualisierung Ergebnis derzeit keine Legende vorhanden ist. Dies ist darauf zurückzuführen, dass die Legende im finalen System eingefügt wird, wie in Abschnitt 6.5 Übergabe an die Fachabteilung beschrieben. In dieser Projektarbeit ist die Legende - bestehend aus Materialnummer, Menge, Preis in Euro, Planlieferzeit in Tagen, der prozentualen Veränderung des Preises anhand der letzten `n` Bestellungen und der prozentualen Veränderung der Planlieferzeit anhand der letzten `n` Bestellungen - noch nicht in der aktuellen Grafik enthalten. Dies stellt jedoch keinen Fehler dar, sondern ist ein geplanter Aspekt des Projektdesigns und wird in der Endphase des Projekts umgesetzt.

5.3 Probleme und Herausforderungen

Die Analyse erfordert aufgrund ihrer hohen Leistungsanforderungen, wie in Abschnitt 3.1 Ressourcenplanung und Recherchen beschrieben, den Einsatz neuer Hardware.

Es ist zu beachten, dass die Performance der Analyse je nach Struktur der Eingabedaten variieren kann. Im schlimmsten Fall könnte die Performance bei O(n^2) liegen, während sie im besten Fall O(n * log(n)) erreicht. Obwohl die Speichernutzung optimiert wurde, benötigt die Ausführung immer noch einige Sekunden. Die durchschnittliche Big O-Leistung der Analyse liegt dabei bei O(n * log(n)), was darauf hinweist, dass die Performanz in Bezug auf die Dateneingabe und die Komplexität der Verarbeitungsschritte skaliert.

6 Projektabschluss

6.1 Wirtschaftlichkeit

Das Projekt ist wirtschaftlich, da für die Analyse und Visualisierung von Druckmaschinen Strukturstücklisten und deren Parameter bisher keine Standardsoftware entwickelt wurde. Das Projekt ist ein erster Vorstoß, um einen Überblick über die bisher gesammelten Rohdaten zu bekommen. Für solche Aufgaben braucht es einzelne Spezialisten von allen Aggregatstypen, die sich sehr gut mit den Daten auskennen. Zudem müssten sie sich die Rohdaten selbst herholen. Dies würde sehr viele Arbeitsstunden in Anspruch nehmen und die strategischen Einkäufer zusätzlich von anderer Arbeit abhalten. Mit der Analyse- und Visualisierungslösung wurde eine kostengünstige und leicht bedienbare Brücke zwischen den Rohdaten und ihrer Auswertung geschaffen, womit man noch nicht zuordenbare Daten innerhalb von Sekunden einschätzen kann.

6.2 Projektkosten

Für die Planung und eine Einführung in Form der Ist-Analyse wurden mehrere Stunden an Besprechungen mit dem Projektverantwortlichen in Anspruch genommen. Deshalb werden diese in die Kostenkalkulation mitberechnet. Zusätzlich wurde das Programm mehrere Stunden mit einem Experten entwickelt, was ebenfalls in der Kalkulation berücksichtigt wird. Die Preise für die Arbeitsstunden stammen aus dem Controlling, daher sind Gemeinkosten miteinberechnet.

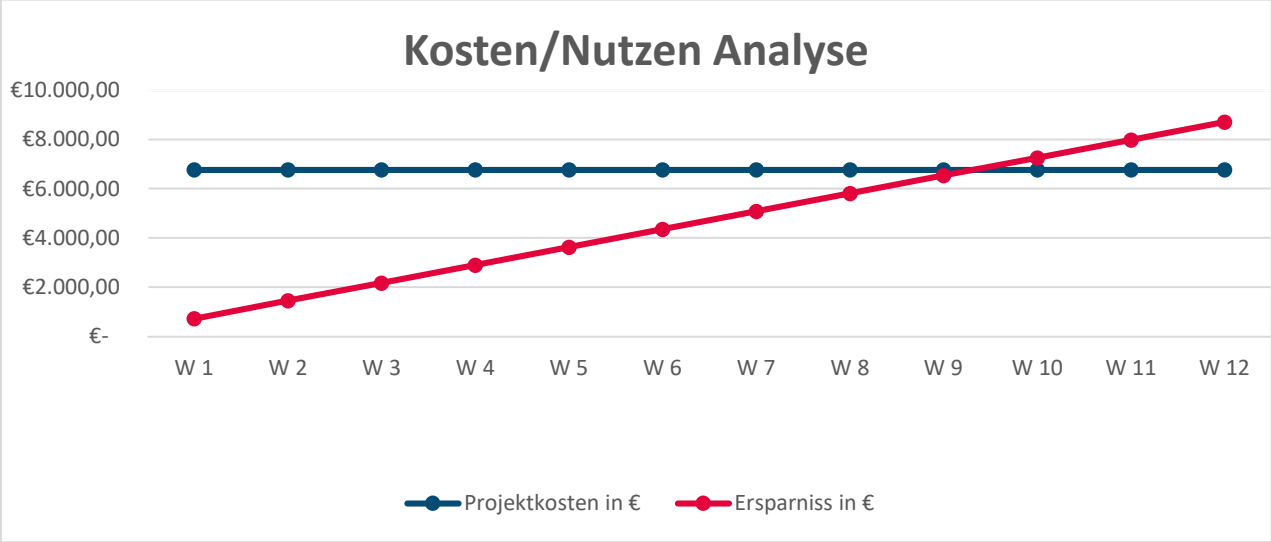
Neben den Arbeitsstunden wurden auch die Kosten für zwei neue Laptops berücksichtigt (siehe dazu: 3.1 Ressourcenplanung und Recherche), die sowohl für mich als auch für den Projektleiter angeschafft wurden. Diese Investition war notwendig, um die technischen Anforderungen und Herausforderungen aus 5.3 Probleme und Herausforderungen des Projekts zu erfüllen und eine effiziente Arbeit zu ermöglichen. Die Kosten für die beiden Laptops sind ebenfalls in der Gesamtkalkulation enthalten.

Position	Preis/h	Menge	Gesamtpreis
Arbeitsstunde Auszubildender	16,54 €	80	1.323,20 €
Arbeitsstunde Projektverantwortlicher	107,50 €	20	2.150,00 €
Arbeitsstunde Experte	89,46 €	10	894,60 €
Fujitsu Lifebook E Series	1.200,00 €	2	2.400,00 €
Gesamtkosten			**6.767,80 €**

Tabelle 3: Projektkostenkalkulation

6.3 Kosten / Nutzen Analyse

Die Gesamtkosten des Projekts betragen 6.767,80 € (siehe: 6.2 Projektkosten), einschließlich der zwei Laptops. Durch das Programm sparen sechs Einkäufer wöchentlich jeweils 1,5 Stunden, was bei einem Stundensatz von 80,65 € pro Einkäufer einer wöchentlichen Gesamteinsparung von 725,85 € entspricht. Die Amortisationszeit des Projekts beträgt daher etwa 9,32 Wochen. Die Kosten-Nutzen-Analyse zeigt, dass das Projekt innerhalb von 9,32 Wochen wirtschaftlich rentabel wird, da es zu einer effizienteren Arbeitsweise und verbessertem Kostenmanagement beiträgt. Zudem hat das Projekt einen nicht abbildbaren imensen Nutzen, da die Mitarbeiter im Strategischen Einkauf einen besseren Überblick über kritische Teile erhalten, und so strukturierter und einfacher gegensteuern können.



6.4 Zeitkorrektur

Der geplante Zeitraum von 80 Stunden konnte eingehalten werden. Jedoch mussten bei der zeitlichen Planung der einzelnen Projektphasen Änderungen vorgenommen werden. Die Ist-Analyse konnte aufgrund der schnellen Einarbeitung in die vorhandenen Daten- und IT-Strukturen schneller als geplant abgeschlossen werden. Die Realisierung des Projektes hat aufgrund einer benötigten Lernphase in das Composite Pattern und die OOP von Python3 zwei Stunden mehr gedauert.

Projektphase	Soll	Ist	Differenz
Ist-Analyse	7,5	5	-2,5
Projektplanung	12,5	12,5	0
Projektrealisierung	46	48	+2
Tests und Fehlerbehebungen	4	2	-2
Projektdokumentation	10	12,5	+2,5
Gesamtdauer	**80**	**80**	**0**

Tabelle 4: Reale Durchführungsdauer

6.5 Übergabe an die Fachabteilung

Nach der erfolgreichen Fertigstellung des Projekts wurde die gesamte Code-Basis an den Projektleiter und die zuständige Fachabteilung übergeben. Der Projektleiter prüfte den Code und führte einen Test durch, um sicherzustellen, dass die Analyse ordnungsgemäß funktioniert. An diesem Punkt wurde auch die bislang fehlende Legende, die aus der Materialnummer, Menge, Preis in Euro, Planlieferzeit in Tagen, der prozentualen Veränderung des Preises und der prozentualen Veränderung der Planlieferzeit, in die Visualisierung eingefügt. Nach der erfolgreichen Validierung und dem Testlauf wurde die Analyse in das Celonis-System integriert, um den Mitarbeitern der Fachabteilung Zugang zu den gewonnenen Informationen und Visualisierungen zu ermöglichen. Damit wurde sichergestellt, dass die Darstellungen und Grafiken vollständig und benutzerfreundlich sind, einschließlich der nun integrierten Legende für eine klare und einfache Dateninterpretation.

6.6 Fazit

Das Projekt stellte anfangs aufgrund der komplexen Anforderungen an die Analyse und Visualisierung von Druckmaschinen Strukturstücklisten und deren Parametern eine Herausforderung dar. Dennoch konnte diese erfolgreich bewältigt werden. Das Python-Programm ist nun im Einsatz und erfüllt alle geplanten Aufgaben. Dies ist der erste Schritt von der MGWS in das noch unbekannte Gebiet der Datenanalyse und Visualisierung im Zusammenhang mit Druckmaschinen.

I. Anlagen

A. manroland Goss Firmengebäude

Abbildung A: MGWS-Firmenzentrale in Augsburg

B. VARIOMAN

Abbildung B: VARIOMAN f:line: Verpackungsdruckmaschine für Plastikfolie und Kartonverpackungen

C. UML-Klassendiagramm

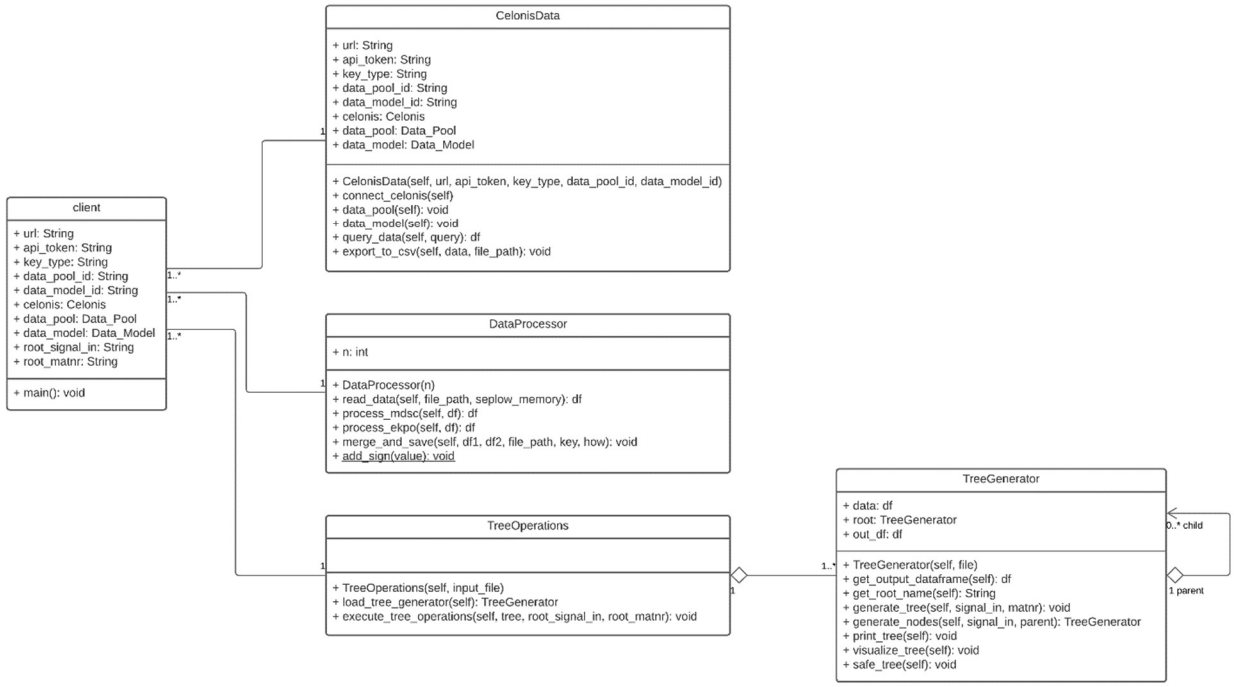


Abbildung C: UML-Klassendiagramm des Analyseprogramms

D. Kompositum Entwurfsmuster - (Composite Pattern)

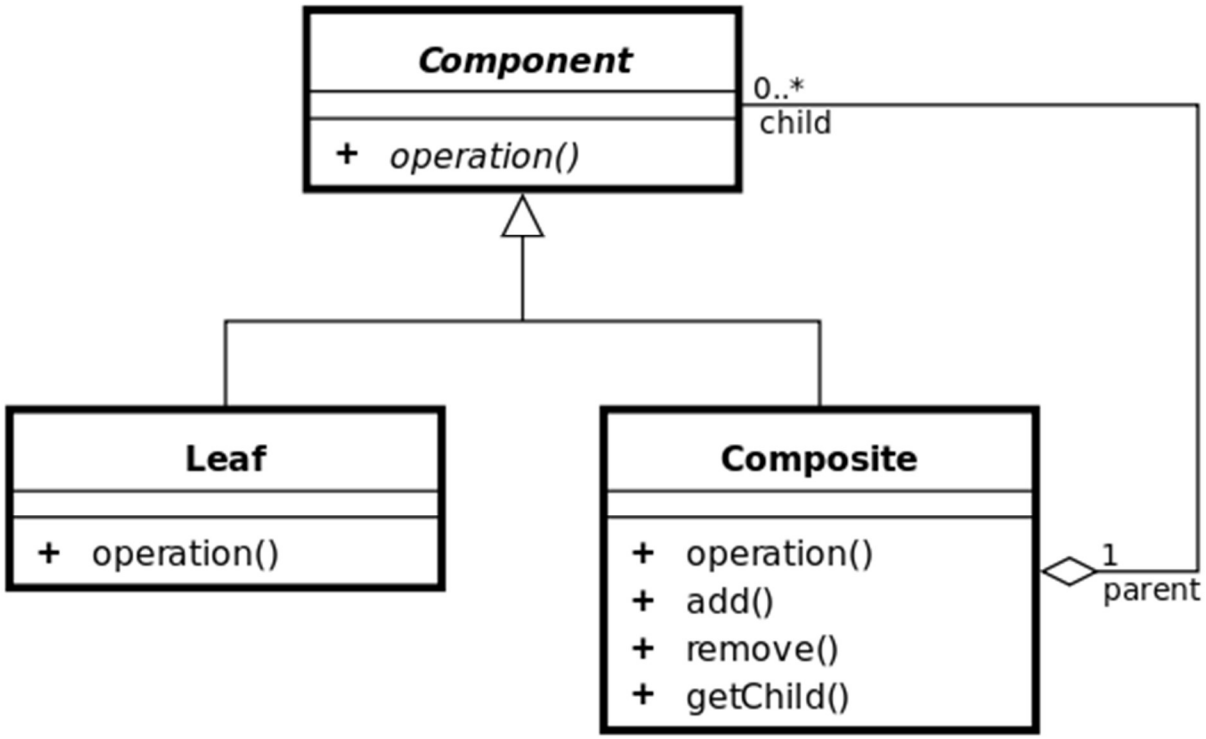


Abbildung D: Kompositum Entwurfsmuster - (Composite Pattern)

E. Dateiverzeichnis Struktur

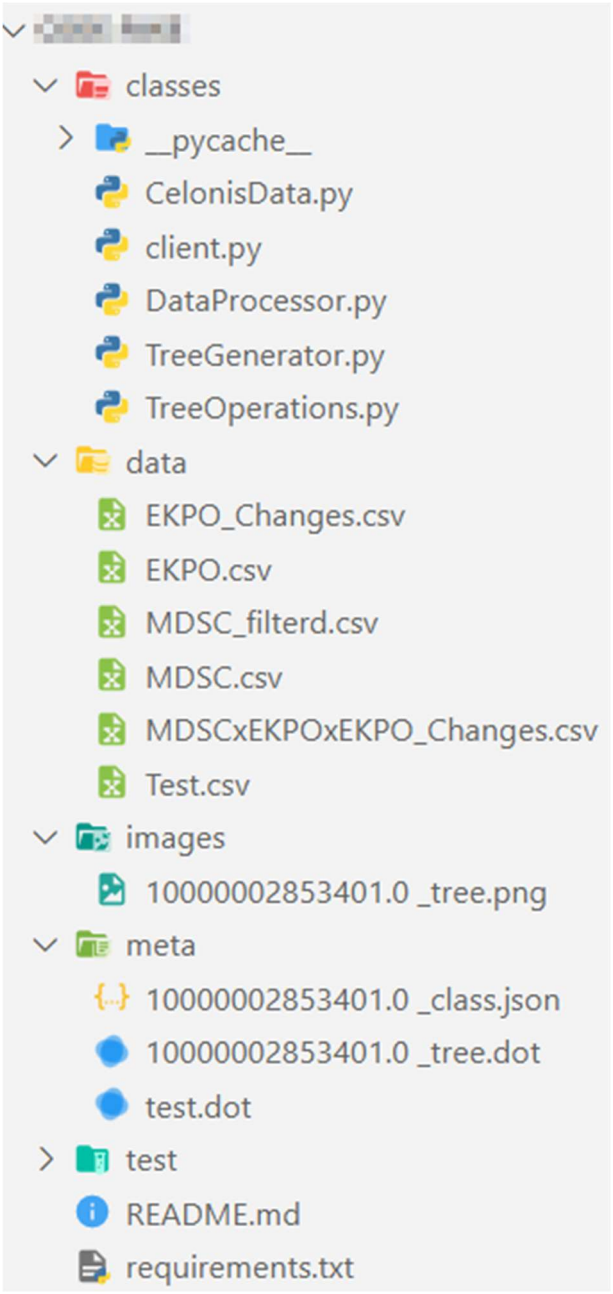


Abbildung E: Dateiverzeichnis mit der Struktur des Codes und deren Daten | VS-Code

F. CelonisData.py

```python
# Importiere benötigte Bibliotheken
import pycelonis
from pycelonis import *
from pycelonis.pql import pql, PQLColumn, PQLFilter, OrderByColumn
import pandas as pd

# Definiere die Klasse CelonisData
class CelonisData:
    # Konstruktor-Methode, die die Initialisierung des Objekts ermöglicht
    def __init__(self, url, api_token, key_type, data_pool_id, data_model_id):
        # Initialisiere die Instanzvariablen
        self.url = url
        self.api_token = api_token
        self.key_type = key_type
        self.data_pool_id = data_pool_id
        self.data_model_id = data_model_id
        self.celonis = None
        self.data_pool = None
        self.data_model = None

    # Methode zum Herstellen einer Verbindung zu Celonis
    def connect_celonis(self):
        self.celonis = get_celonis(base_url=self.url, api_token=self.api_token, key_type=self.key_type)

    # Methode zum Abrufen des Datenpools
    def get_data_pool(self):
        self.data_pool = self.celonis.data_integration.get_data_pool(self.data_pool_id)

    # Methode zum Abrufen des Datenmodells
    def get_data_model(self):
        self.data_model = self.data_pool.get_data_model(self.data_model_id)

    # Methode zum Ausführen einer Abfrage und Rückgabe der Ergebnisse als DataFrame
    def query_data(self, query):
        query_result = self.data_model.export_data_frame(query)
        return query_result

    # Methode zum Exportieren der Ergebnisse in eine CSV-Datei
    def export_to_csv(self, data, file_path):
        data.to_csv(file_path, index=False)
```

Abbildung F: CelonisData.py | Klassen-Programmcode zur Aggregation von Daten aus dem Celonis-EMS | Python3

G. DataProcessor.py

```python
# Importiere benötigte Bibliotheken
import pandas as pd
import numpy as np

# Definiere die Klasse DataProcessor
class DataProcessor:
    # Konstruktor-Methode
    def __init__(self, n):
        self.n = n

    # Methode zum Lesen von Daten aus einer CSV-Datei
    def read_data(self, file_path, sep=',', low_memory=False):
        return pd.read_csv(file_path, sep=sep, low_memory=low_memory)

    # Methode zur Verarbeitung der MDSC-Daten
    def process_mdsc(self, df):
        df.set_index(["SIGNAL_IN","SIGNAL_OUT","MATNR","EVENTTIME","EVENTTIME_T0","MENGE_OUT"])
        df = df[pd.isnull(df['EVENTTIME_T0'])]
        df = df.drop('EVENTTIME', axis=1)
        df = df.drop('EVENTTIME_T0', axis=1)
        df = df.replace(np.nan, "NaN", regex=True)
        return df

    # Methode zur Verarbeitung der EKPO-Daten
    def process_ekpo(self, df):
        df['_CELONIS_CHANGE_DATE'] = pd.to_datetime(df['_CELONIS_CHANGE_DATE'])
        df = df.sort_values(['MATNR', '_CELONIS_CHANGE_DATE'], ascending=[True, False])
        group_id = df.groupby('MATNR').cumcount()
        last_n_df = df[group_id < self.n]
        agg_df = last_n_df.groupby('MATNR').agg({'NETWR_CONVERTED': 'mean', 'PLIFZ': 'mean'})
        agg_df['NETWR_PCT_CHANGE'] = agg_df['NETWR_CONVERTED'].pct_change() * 100
        agg_df['PLIFZ_PCT_CHANGE'] = agg_df['PLIFZ'].pct_change() * 100
        agg_df['NETWR_PCT_CHANGE'] = agg_df['NETWR_PCT_CHANGE'].apply(self.add_sign)
        agg_df['PLIFZ_PCT_CHANGE'] = agg_df['PLIFZ_PCT_CHANGE'].apply(self.add_sign)
        agg_df = agg_df.round(2)
        agg_df.fillna(0.00, inplace=True)
        return agg_df

    # Methode zum Zusammenführen von zwei DataFrames und Speichern in einer CSV-Datei
    def merge_and_save(self, df1, df2, file_path, key, how="left"):
        merged_df = pd.merge(df1, df2, on=key, how=how)
        merged_df.to_csv(file_path, index=False)

    # Statische Methode zum Hinzufügen von Vorzeichen (+/-) zu numerischen Werten
    @staticmethod
    def add_sign(value):
        if value > 0:
            return f"+{value:.2f}"
        elif value < 0:
            return f"{value:.2f}"
        else:
            return f"{value:.2f}"
```

Abbildung G: DataProcessor.py | Klassen-Programmcode zum Filtern von Daten, die aus dem Celonis-EMS aggregiert wurden | Python3

H. TreeOperations.py

```python
# Importiere benötigte Bibliotheken
import pandas as pd
from TreeGenerator import TreeGenerator

# Definiere die Klasse TreeOperations
class TreeOperations:
    # Konstruktor-Methode
    def __init__(self, input_file):
        self.input_file = input_file

    # Methode zum Laden des TreeGenerator-Objekts
    def load_tree_generator(self):
        tree = TreeGenerator(self.input_file)
        return tree

    # Methode zum Ausführen von Baumoperationen
    def execute_tree_operations(self, tree, root_signal_in, root_matnr):
        # Zeige den geladenen DataFrame
        print("Load OG DF: DONE")
        print(tree.data.head)

        # Zeige die Wurzelinformationen
        print("Root Signal_In: " + root_signal_in + "\nRoot MATNR: " + root_matnr)
        print("\n \n ---------------------------------------------------------------- \n \n")

        # Generiere den Baum und zeige die Baumstruktur
        tree.generate_tree(root_signal_in, root_matnr)
        print("Generate Tree: DONE")
        print(tree.print_tree())
        print("\n \n ---------------------------------------------------------------- \n \n")

        # Erhalte den DataFrame des Baums und zeige ihn an
        print("Tree DF: DONE")
        output_tree_df = tree.get_output_dataframe()
        print(output_tree_df.head)
        print("\n \n ---------------------------------------------------------------- \n \n")

        # Baum speichern und visualisieren
        tree.safe_tree()
        tree.visualize_tree()
        print("Visualize and Save: DONE")
        print("\n \n ---------------------------------------------------------------- \n \n")
```

Abbildung H: Auszug von TreeOperations.py | Klassen-Programmcode zum Erzeugen der Datenstruktur und Visualisierung| Python3

I. TreeGenerator.py (Auszug)

```
...

    # Getter & Setter
    def get_output_dataframe(self):
        """hole outputdataframe
        """
        return self.out_df

    def get_root_name(self):
        return self.root.name[0]

    #! Baum und Generator Algorythmus
    # Generate Tree
    def generate_tree(self, signal_in, matnr):
        """Funktion zur Erzeugung des gesamten Baums auf der Grundlage eines Eingangssignals und matnr
        """
        # get entry in the dataframe
        signal_in_data = self.data.loc[((self.data['SIGNAL_IN']==signal_in) & (self.data['MATNR']==matnr))]
        # if entry exists
        if len(signal_in_data)>0:
            # create first row with root entry
            new_row={"SIGNAL_IN": signal_in_data.values.tolist()[0][0],
                     "SIGNAL_OUT": signal_in_data.values.tolist()[0][1] ,
                     "MATNR": signal_in_data.values.tolist()[0][2],
                     "MENGE_OUT":signal_in_data.values.tolist()[0][3],
                     "NETWR_CONVERTED":signal_in_data.values.tolist()[0][4],
                     "PLIFZ":signal_in_data.values.tolist()[0][5],
                     "NETWR_PCT_CHANGE":signal_in_data.values.tolist()[0][5],
                     "PLIFZ_PCT_CHANGE":signal_in_data.values.tolist()[0][5]
                    }

            # create root node
            self.root = Node(signal_in_data.values.tolist()[0])
            # append row to output dataframe
            new_df = pd.DataFrame(data=new_row,index=[0])
            self.out_df=pd.concat([self.out_df,new_df], ignore_index=True)
            # go on with other nodes of the tree
            self.generate_nodes(signal_in, self.root)
        else:
            # if no entry, there can't be generated a tree
            print("No object with these parameters!")

    # Generate Nodes / Leafes
    def generate_nodes(self, signal_in, parent):
        """Funktion zur Erzeugung aller möglichen Knotenpunkte für ein Eingangssignal
        """
        # alle Daten ermitteln, bei denen signal_in == signal_out
        signal_out_data = self.data.loc[(self.data['SIGNAL_OUT']==signal_in)]
        # wenn Knoten existieren
        if len(signal_out_data)>0:
            # Liste für alle Signal_in_nodes erstellen
            signal_in_nodes=[]
            # Iteration über alle signal_out-Kinderknoten
            for elem in zip(signal_out_data['SIGNAL_IN'], signal_out_data['SIGNAL_OUT'],
signal_out_data['MATNR'],signal_out_data['MENGE_OUT']):
                # von Tupel in Liste umwandeln
                elem = list(elem)
                # Eintrag für output_dataframe erstellen
                new_row={"SIGNAL_IN": elem[0], "SIGNAL_OUT": elem[1] , "MATNR": elem[2] ,"MENGE_OUT":elem[3]}
                # neuen Knoten mit übergeordnetem Knoten aus Parametern erstellen
                node = Node(elem, parent=parent)
                # Zeile in dataframe einfügen
                new_df = pd.DataFrame(data=new_row,index=[0])
                self.out_df=pd.concat([self.out_df,new_df], ignore_index=True)
                # Signal_in-Daten mit derselben Matrize wie Signal_out_data abrufen
                signal_in_data = self.data.loc[((self.data['MATNR'] == elem[2]) & (self.data['SIGNAL_IN']!='nan'))]
                # wenn vorhanden
                if len(signal_in_data)>0:
                    # Signal_in Kind_Knoten erstellen
                    signal_in_node = Node(signal_in_data.values.tolist()[0], parent=node)
                    # an die Liste signal_in_nodes_list anhängen
                    signal_in_nodes.append(signal_in_node)
                else:
                    # sonst weiter mit nächstem Signal aus Kindknoten
                    #print("No child nodes any more!")
                    pass
            # wenn es signal_in Kindknoten gibt
            if len(signal_in_nodes) >0:
                # Iteration über signal_in-Kinderknoten
                for n in signal_in_nodes:
                    # neuen Eintrag erst jetzt erstellen wegen der Reihenfolge im Datenrahmen (Beispiel: root--
child1--child2--child1.child1--child1.child2)
                    new_row_signal_in={"SIGNAL_IN": n.name[0], "SIGNAL_OUT": n.name[1] , "MATNR": n.name[2]
,"MENGE_OUT":n.name[3]}
                    # Einfügen in den Ausgabedatenrahmen
                    new_signal_in_df = pd.DataFrame(data=new_row_signal_in,index=[0])
                    self.out_df=pd.concat([self.out_df,new_signal_in_df], ignore_index=True)
                    # die Funktion rekursiv aufrufen, bis keine Kindknoten mehr vorhanden sind
                    self.generate_nodes(n.name[0], n)
            else:
                # Keine signal_in-Kinderknoten
                return
        else:
            # Keine signal_out-Kinderknoten
            print("Object has no child nodes!")
            return
```

```
...

    #! Ausgabe & Visualisierung
    #* Print Tree
    def print_tree(self,):
        """ Baum im Terminal ausdrucken
        """
        for pre, fill, node in RenderTree(self.root):
                print("%s%s" % (pre, node.name))

    #* Visualize Tree
    def visualize_tree(self):
            """Visualisierung als png speichern
            """
            DotExporter(self.root).to_picture(str("images/" + self.root.name[0]) + " _tree.png")

    #* Save Tree
    def safe_tree(self):
        """Visualisierung als png speichern mit Grappviz"""

        path = str("meta/" + self.root.name[0]) + " _tree.dot"

        dot_exporter = UniqueDotExporter(self.root)
        dot_exporter.to_dotfile(path)
```

Abbildung I: TreeGenerator.py | Klassen-Programmcode zum Erzeugen eines Composite Pattern und Ausgabe von Attributen der Daten | Python3

J. client.py

```python
#from classes.TreeGenerator import TreeGenerator
from CelonisData import CelonisData
from DataProcessor import DataProcessor
from TreeOperations import TreeOperations

import pycelonis
from pycelonis import *
from pycelonis.pql import pql, PQLColumn, PQLFilter, OrderByColumn

import pandas as pd
import numpy as np

if __name__ == '__main__':
    url = "https://manroland.eu-5.celonis.cloud"
    api_token = "censored"
    key_type = "censored"

    celonis_data = CelonisData(url, api_token, key_type, "censored", "censored")
    celonis_data.connect_celonis()
    celonis_data.get_data_pool()
    celonis_data.get_data_model()

    # Query MDSC
    query_MDSC = pql.PQL()
    query_MDSC += pql.PQLColumn(name="SIGNAL_IN", query=""" "MDSC"."SIGNAL_IN" """)
    query_MDSC += pql.PQLColumn(name="SIGNAL_OUT", query=""" "MDSC"."SIGNAL_OUT" """)
    query_MDSC += pql.PQLColumn(name="MATNR", query=""" "MDSC"."MATNR" """)
    query_MDSC += pql.PQLColumn(name="EVENTTIME", query=""" "MDSC"."EVENTTIME" """)
    query_MDSC += pql.PQLColumn(name="EVENTTIME_T0", query=""" "MDSC"."EVENTTIME_T0" """)
    query_MDSC += pql.PQLColumn(name="MENGE_OUT", query=""" "MDSC"."MENGE_OUT" """)
    result_MDSC = celonis_data.query_data(query_MDSC)
    celonis_data.export_to_csv(result_MDSC, 'data/MDSC.csv')

    # Query EKPO
    query_EKPO = pql.PQL()
    query_EKPO += pql.PQLColumn(name="_CASE_KEY", query="""EKPO"."_CASE_KEY" """)
    query_EKPO += pql.PQLColumn(name="_CELONIS_CL_TIMESTAMP", query=""" "EKPO"."_CELONIS_CL_TIMESTAMP" """)
    query_EKPO += pql.PQLColumn(name="_CELONIS_CHANGE_DATE", query=""" "EKPO"."_CELONIS_CHANGE_DATE" """)
    query_EKPO += pql.PQLColumn(name="MATNR", query=""" "EKPO"."MATNR" """)
    query_EKPO += pql.PQLColumn(name="EMATN", query=""" "EKPO"."EMATN" """)
    query_EKPO += pql.PQLColumn(name="MATNR_TEXT", query=""" "EKPO"."MATNR_TEXT" """)
    query_EKPO += pql.PQLColumn(name="MATKL_TEXT", query=""" "EKPO"."MATKL_TEXT" """)
    query_EKPO += pql.PQLColumn(name="NETWR_CONVERTED", query=""" "EKPO"."NETWR_CONVERTED" """)
    query_EKPO += pql.PQLColumn(name="WAERS", query=""" "EKPO"."WAERS" """)
    query_EKPO += pql.PQLColumn(name="MATKL", query=""" "EKPO"."MATKL" """)
    query_EKPO += pql.PQLColumn(name="WERKS", query=""" "EKPO"."WERKS" """)
    query_EKPO += pql.PQLColumn(name="PLIFZ", query=""" "EKPO"."PLIFZ" """)
    result_EKPO = celonis_data.query_data(query_EKPO)
    celonis_data.export_to_csv(result_EKPO, 'data/EKPO.csv')


    data_processor = DataProcessor(n=3)

    df_MDSC = data_processor.read_data("data/MDSC.csv")
    df_MDSC = data_processor.process_mdsc(df_MDSC)
    df_MDSC.to_csv('data/MDSC_filterd.csv', index=False)

    df_EKPO = data_processor.read_data("data/EKPO.csv")
    agg_df = data_processor.process_ekpo(df_EKPO)
    agg_df.to_csv("data/EKPO_Changes.csv")

    EKPOxEKPO_Changes_df = pd.merge(df_EKPO, agg_df, on="MATNR", how="right")
    data_processor.merge_and_save(df_MDSC, EKPOxEKPO_Changes_df, "data/MDSCxEKPOxEKPO_Changes.csv", "MATNR")


    tree_operations = TreeOperations('MDSCxEKPOxEKPO_Changes.csv')
    tree_gen = tree_operations.load_tree_generator()

    root_signal_in = "10000002853401.0"
    root_matnr = "16040003410"

    tree_operations.execute_tree_operations(tree_gen, root_signal_in, root_matnr)
```

Abbildung J: client.py | Programm Code zur Aggregation der Daten und Erstellung der Analyse und Visualisierung mit dem Beispiel an der MATNR: 16040003410

K. ZSB.FARB- FEUCHTWALZEN

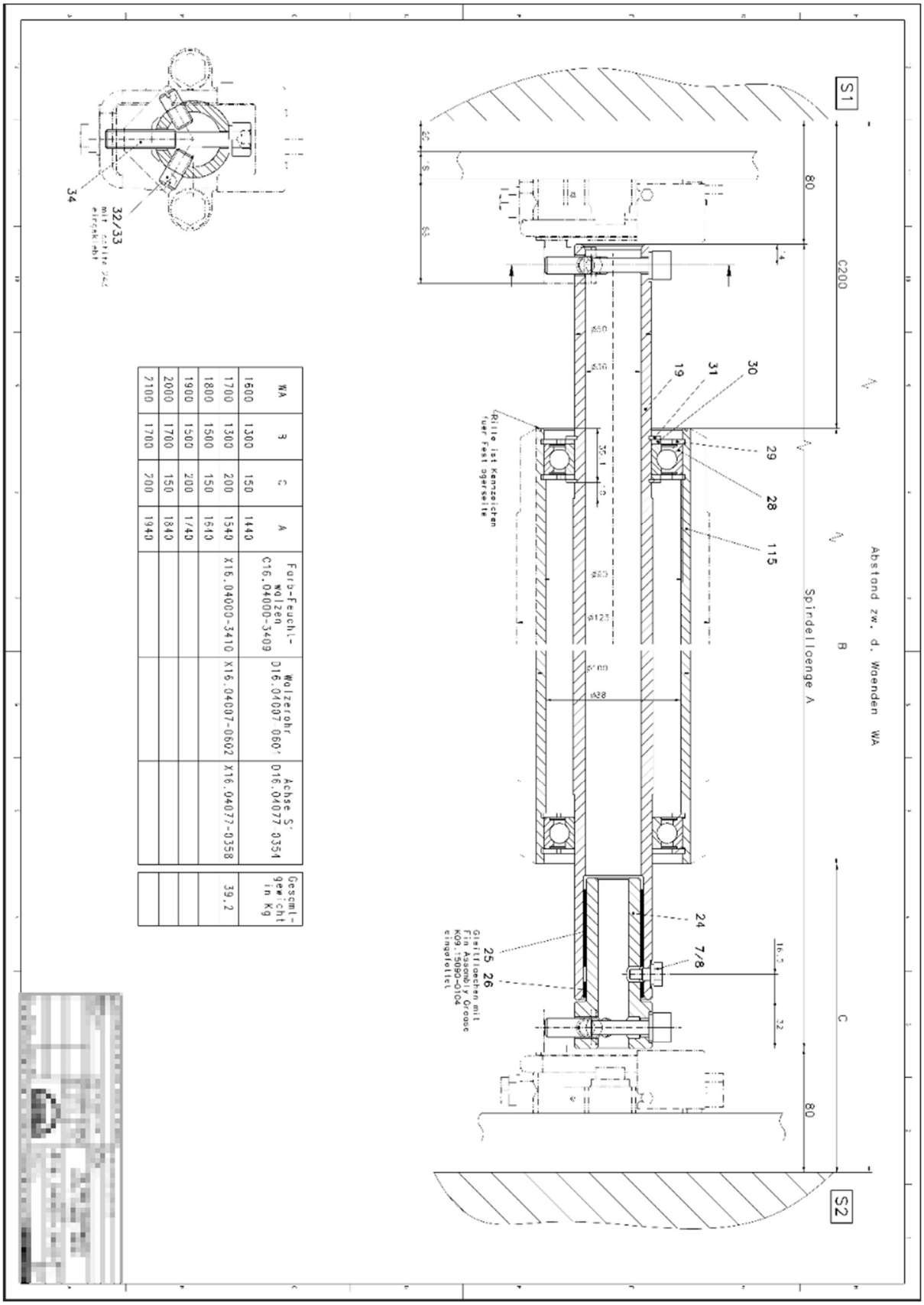


Abbildung K: ZSB.FARB- FEUCHTWALZEN WA=1700, B=1300, MATNR: 16040003410 | Feuchtwalze zur Übertragung von Farbe vom Farbkasten auf das Papier | Zentrale Komponente eines Druckwerks

L. Test-Daten (Auszug)

data > [redacted]

```
SIGNAL_IN,SIGNAL_OUT,MATNR,MENGE_OUT,NETWR_CONVERTED,PLIFZ,NETWR_PCT_CHANGE,PLIFZ_PCT_CHANGE
10000002853401,NaN,16040003410,0.0,0.0,0.0,0%,0%
NaN,10000002853401,06353203525,2.0,0.29,5,+2%,-12%
NaN,10000002853401,06151134134,2.0,3.11,12,+20%,+10%
NaN,10000002853401,06290100135,2.0,3.45,43,+23%,+7%
NaN,10000002853401,06314290208,2.0,5.20,26,+10%,+50%
NaN,10000002853401,16041940086,2.0,0.3,23,0%,0%
NaN,10000002853401,06012831111,1.0,5.66,32,+120%,-20%
NaN,10000002853401,06353203510,1.0,0.23,34,0%,+12%
NaN,10000002853401,09230000243,1.0,24.99,7,0%,0%
NaN,10000002853401,06060401513,4.0,70.23,103,+221%,+103%
NaN,10000002853401,06290200208,4.0,32.59,34,+45%,-2%
NaN,10000002853401,16040070602,1.0,1012.00,12,+30%,+80%
NaN,10000002853401,16040770358,1.0,420.50,234,+45%,+304%
NaN,10000002853401,16040940025,0.0,0.0,0.0,0%,0%
NaN,10000002853401,16040770367,0.0,0.0,0.0,0%,0%
10000035297901,NaN,16040940025,0.0,0.0,0.0,0%,0%
NaN,10000035297901,06012831111,1.0,5.66,32,+120%,-20%
10000031345201,NaN,06353203525,0.0,,0.0,0.0,0%,0%
NaN,10000031345201,06353203530,1.0,0.29,5,+2%,-12%
10000035165802,NaN,16040770358,0.0,0.0,0.0,0%,0%
NaN,10000035165802,16889105127,1.0,4.23,10,+20%,-23%
10000035188101,NaN,16040070602,0.0,3.23,12,+40%,+53%
NaN,10000035188101,16889104449,1.0,4.53,50,+27%,+23%
10000035166401,NaN,16040770367,0.0,0.0,0.0,0%,0%
NaN,10000035166401,16889012862,1.0,1.23,34,0%,+11%
```

Abbildung L: Auszug der Test-Daten einer .CSV-Datei zum Testen des Programms als Eingabeparameter für den Client

M. Programmausgabe 1

```
------------------------------------------------------------

Generate Tree: DONE
["10000002853401","nan","16040003410","0.0","0.0","0.0","0.0","0.0"]
├── ["10000031345201","nan","06353203525","0.0",,"0.0","0.0","0.0","0.0"]
│   └── ["nan","10000002853401","06353203525","2.0","0.29","5","+2%","-12%"]
│       └── ["nan","10000031345201","06353203530","1.0","0.29","5","+2%","-12%"]
├── ["nan","10000002853401","06151134134","2.0","3.11","12","+20%","+1.0"]
├── [Na"N,"10000002853401","06290100135","2.0","3.45","43","+23%","+7%"]
├── ["nan","10000002853401","06314290208","2.0","5.20","26","+10%","+50%"]
├── ["nan","10000002853401","16041940086","2.0","0.3","23","0.0","0.0"]
├── ["nan","10000035297901","06012831111","1.0","5.66","32","+12%","-2%"]
├── ["nan","10000002853401","06353203510","1.0","0.23","34","0.0","+12%"]
├── ["nan","10000002853401","09230000243","1.0","24.99","7","0.0","0.0"]
├── ["nan","10000002853401","06060401513","4.0","70.23","103","+221%","+103%"]
├── ["nan","10000002853401","06290200208","4.0","32.59","34"",+45%","-2%"]
├── ["nan","10000002853401","16040070602","1.0","1012.00","60","+30%","+80%"]
│   └── ["10000035188101","nan","16040070602","0.0","0.0","0.0","0.0","0.0"]
│       └── ["nan","10000035188101","16889104449","1.0","4.53","50","+27%","+23%"]
├── ["nan","10000002853401","16040770358","1.0","420.50","234","+45%","+304%"]
│   └── ["10000035165802","nan","16040770358","0.0","0.0","0.0","0.0","0.0"]
│       └── ["nan","10000035165802","16889105127","1.0","4.23","10","+20%","-23%"]
├── ["nan","10000002853401","16040940025","0.0","0.0","0.0","0.0","0.0"]
│   └── ["10000035297901","nan","16040940025","0.0","0.0","0.0","0.0","0.0"]
│       └── ["nan","10000035297901","06012831111","1.0","5.66","32","+120%","-20%"]
└── ["nan","10000002853401","16040770367","0.0","0.0","0.0","0.0","0.0"]
    └── ["10000035166401","nan","16040770367","0.0","0.0","0.0","0.0","0.0"]
        └── ["nan","10000035166401","16889012862","1.0","1.23","34","0.0","+11%"]
None


------------------------------------------------------------
```

Abbildung M: 1. Programmausgabe von client.py der Datenstruktur für die MATNR: 16040003410

N. CSMB-Ausgabe

Produktstruktur	F..	K	Posi...	Menge	K.	Kurztext	S...	E...	I...	I	PosText	PosText
16.04000-3410 1000 2 01						ZSB.FARB- FEUCHTWALZEN WA=1700, B=1300						
0010 T	X		0010	1	ST	Zeichnung 16040003409	000				Zeichnung 16040003409	ZSB.FARB- FEUCHTWALZEN D=1...
0040 X 16.04007-0602			0040	1	ST	WALZENROHR A=1600 B=1300	115	16	00...	1		
16.04007-0602 1000 1 81						WALZENROHR A=1600 B=1300						
9008 L 16.88910-4449	X	X	9008	1	ST	ROHR 100X6XL1304 EN 10305-2 E235+SR TOL.	000	16	08...	2		
16.88910-4449 3000 1 01						ROHR 100X6XL1304 EN 10305-2 E235+SR TOL.						
9001 R 01.94132-9003	X	X	9001	1,304	M	ROHR 100X6 EN 10305-2 E235+SR TOL.F.INNE	000	01	94...	2		
0050 X 16.04077-0358			0050	1	ST	ACHSE S1,WA=1700,L=1508,EBENE2	019	16	00...	1		
16.04077-0358 1000 1 82						ACHSE S1,WA=1700,L=1508,EBENE2						
9008 L 16.88910-5127	X	X	9008	1	ST	ROHR 52X8XL1512 EN 10305-2 E235+C	000	16	08...	2		
16.88910-5127 3000 1 01						ROHR 52X8XL1512 EN 10305-2 E235+C						
9002 R 01.94157-9541	X	X	9002	1,512	M	ROHR 52X8 EN 10305-2 E235+C	000	01	94...	2		
0060 X 16.04077-0367			0060	1	ST	ACHSE S2, IROLOC	024	16	00...	2		
16.04077-0367 1000 1 01						ACHSE S2, IROLOC						
9001 L 16.88901-2862	X	X	9001	1	ST	RUND 55XL111 EN 10060 S355J0+AR	000	16	08...	2		
16.88901-2862 3000 1 01						RUND 55XL111 EN 10060 S355J0+AR						
9007 R 01.12062-0075	X	X	9007	0,111	M	RUND 55 EN 10060 S355J0+AR	000	01	12...	2		
0070 X 06.31429-0208			0070	2	ST	RILLENKUGELLAGER DIN 625 6210 C3 NTN/LLU	028	06	31...	2		
0080 X 06.29020-0208			0080	4	ST	SICHERUNGSRING DIN 472 90X3	029	06	29...	2		
0090 X 06.15113-4134			0090	2	ST	STUETZSCHEIBE DIN 988 S50X62	030	06	15...	2		
0100 X 06.29010-0135			0100	2	ST	SICHERUNGSRING DIN 471 50X2	031	06	29...	2		
0110 X 06.06040-1513			0110	4	ST	SCHAFTSCHRAUBE DIN 427 M12X25 14H A2B;TO	032	06	06...	2		
0120 X 09.23000-0243			0120	1	ST	SCHRAUBENSICHERUNG LOCTITE 243 50 ML HEN	033	09	23...	2		
0130 X 16.04194-0086			0130	2	ST	SCHRAUBE	034	16	00...	2		
0140 X 06.01283-1111			0140	1	ST	6KT-SCHRAUBE DIN 933 M8X16 8.8	006	06	01...	2		
0150 X 16.04094-0025			0150	1	ST	SECHSKANTSCHRAUBE	007	16	00...	2		
0160 X 06.35320-3525			0160	2	ST	BUCHSE ISO 3547 35B39X25 P1	025	06	35...	2		
0170 X 06.35320-3510			0170	1	ST	BUCHSE ISO 3547 35B39X10 P1	026	06	35...	2		

Abbildung N: Programmausgabe der SAP-Transaktion CSMB für die MATNR: 16040003410

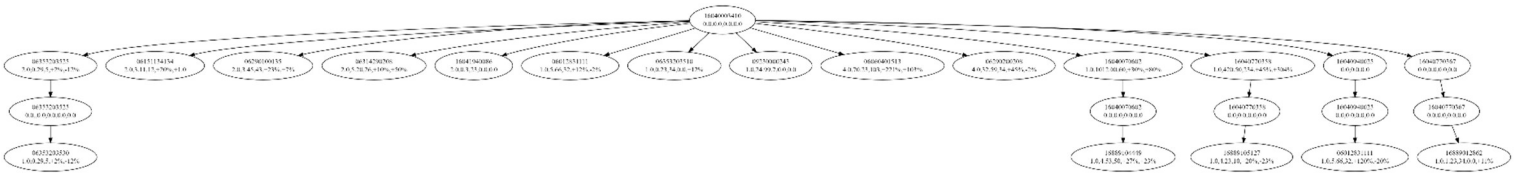
O. Visualisierung Ergebnis Raw

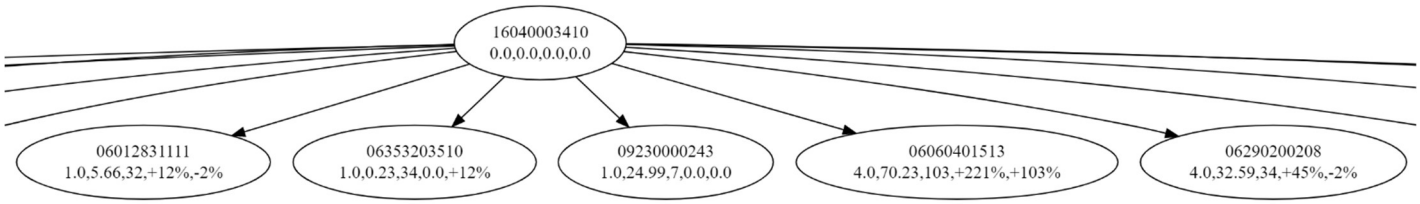
meta > 10000002853401.0 _tree.dot

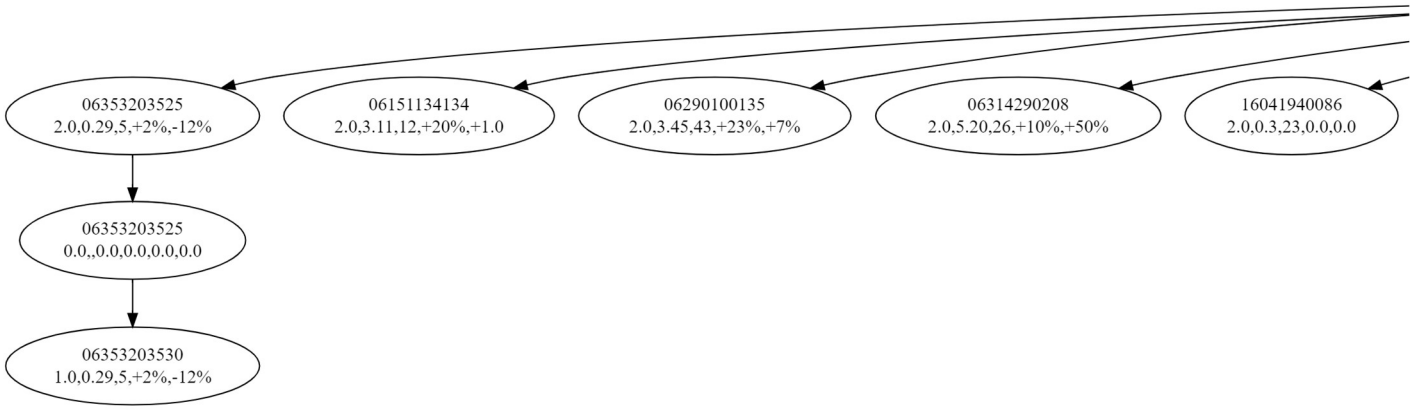
```
digraph tree {
"0x1b264993970" [label="['10000002853401','nan','16040003410','0.0','0.0','0.0','0.0','0.0']"];
"0x1b2649e4760" [label="['nan','10000002853401','06353203525','2.0','0.29','5','+2%','-12%']"];
"0x1b2649e4e50" [label="['10000031345201','nan','06353203525','0.0','','0.0','0.0','0.0','0.0']"];
"0x1b264993880" [label="['nan','10000031345201','06353203530','1.0','0.29','5','+2%','-12%']"];
"0x1b2649e4ee0" [label="['nan','10000002853401','06151134134','2.0','3.11','12','+20%','+1.0']"];
"0x1b2649e4100" [label="['nan','10000002853401','06290100135','2.0','3.45','43','+23%','+7%']"];
"0x1b2649e48b0" [label="['nan','10000002853401','06314290208','2.0','5.20','26','+10%','+50%']"];
"0x1b2649e4970" [label="['nan','10000002853401','16041940086','2.0','0.3','23','0.0','0.0']"];
"0x1b2649e4a90" [label="['nan','10000035297901','06012831111','1.0','5.66','32','+12%','-2%']"];
"0x1b2649e4d90" [label="['nan','10000002853401','06353203510','1.0','0.23','34','0.0','+12%']"];
"0x1b2649e4b80" [label="['nan','10000002853401','09230000243','1.0','24.99','7','0.0','0.0']"];
"0x1b2649e4af0" [label="['nan','10000002853401','06060401513','4.0','70.23','103','+221%','+103%']"];
"0x1b2649e5030" [label="['nan','10000002853401','06290200208','4.0','32.59','34','+45%','-2%']"];
"0x1b2649e4910" [label="['nan','10000002853401','16040070602','1.0','1012.00','60','+30%','+80%']"];
"0x1b2649e49a0" [label="['10000035188101','nan','16040070602','0.0','0.0','0.0','0.0','0.0']"];
"0x1b2649e4ac0" [label="['nan','10000035188101','16889104449','1.0','4.53','50','+27%','+23%']"];
"0x1b2649e4cd0" [label="['nan','10000002853401','16040770358','1.0','420.50','234','+45%','+304%']"];
"0x1b2649e4280" [label="['10000035165802','nan','16040770358','0.0','0.0','0.0','0.0','0.0']"];
"0x1b2649e5d50" [label="['nan','10000035165802','16889105127','1.0','4.23','10','+20%','-23%']"];
"0x1b2649e5060" [label="['nan','10000002853401','16040940025','0.0','0.0','0.0','0.0','0.0']"];
"0x1b2649e48e0" [label="['10000035297901','nan','16040940025','0.0','0.0','0.0','0.0','0.0']"];
"0x1b2649e5750" [label="['nan','10000035297901','06012831111','1.0','5.66','32','+120%','-20%']"];
"0x1b2649e4d00" [label="['nan','10000002853401','16040770367','0.0','0.0','0.0','0.0','0.0']"];
"0x1b2649e4f10" [label="['10000035166401','nan','16040770367','0.0','0.0','0.0','0.0','0.0']"];
"0x1b2649e59c0" [label="['nan','10000035166401','16889012862','1.0','1.23,34','0.0','+11%']"];

    "0x1b264993970" -> "0x1b2649e4760";
    "0x1b264993970" -> "0x1b2649e4ee0";
    "0x1b264993970" -> "0x1b2649e4100";
    "0x1b264993970" -> "0x1b2649e48b0";
    "0x1b264993970" -> "0x1b2649e4970";
    "0x1b264993970" -> "0x1b2649e4a90";
    "0x1b264993970" -> "0x1b2649e4d90";
    "0x1b264993970" -> "0x1b2649e4b80";
    "0x1b264993970" -> "0x1b2649e4af0";
    "0x1b264993970" -> "0x1b2649e5030";
    "0x1b264993970" -> "0x1b2649e4910";
    "0x1b264993970" -> "0x1b2649e4cd0";
    "0x1b264993970" -> "0x1b2649e5060";
    "0x1b264993970" -> "0x1b2649e4d00";
    "0x1b2649e4760" -> "0x1b2649e4e50";
    "0x1b2649e4e50" -> "0x1b264993880";
    "0x1b2649e4910" -> "0x1b2649e49a0";
    "0x1b2649e49a0" -> "0x1b2649e4ac0";
    "0x1b2649e4cd0" -> "0x1b2649e4280";
    "0x1b2649e4280" -> "0x1b2649e5d50";
    "0x1b2649e5060" -> "0x1b2649e48e0";
    "0x1b2649e48e0" -> "0x1b2649e5750";
    "0x1b2649e4d00" -> "0x1b2649e4f10";
    "0x1b2649e4f10" -> "0x1b2649e59c0";
}
```

Abbildung O: Visualisierung der Baugruppe und deren Parametern für die MATNR: 16040003410 in .dot Rohform

P. Visualisierung Ergebnis







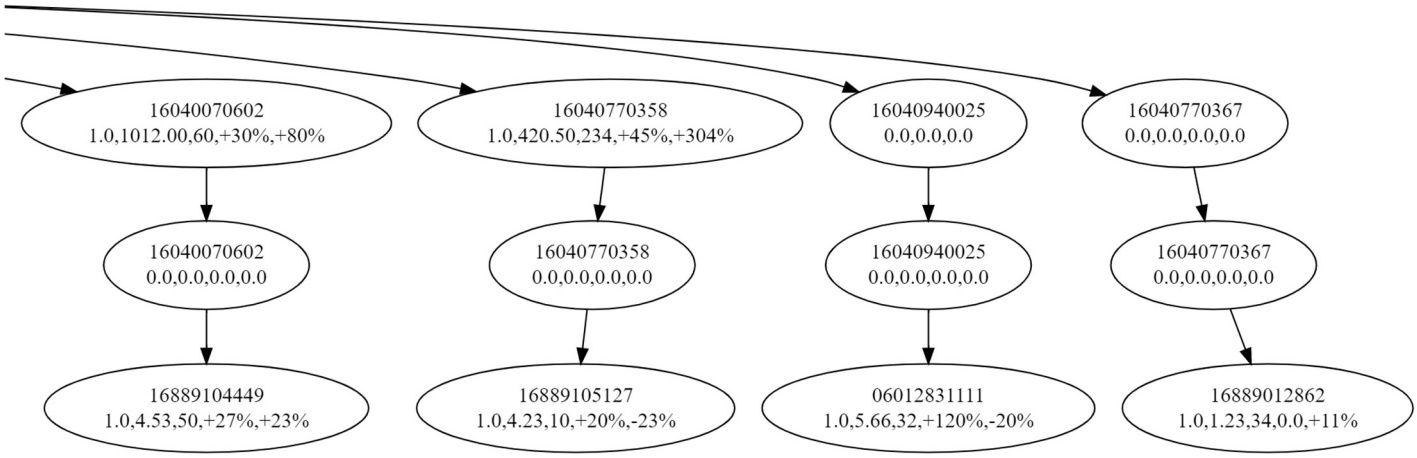


Abbildung P: Visualisierung der Baugruppe und deren Parametern für die MATNR: 16040003410 in .dot | Im Vollformat und zur genaueren Visualisierung in einzelnen Teilen

II. Quellenangaben

Name	Link
Python3	https://www.python.org/
pip3	https://pypi.org/project/pip/
SAP S/4HANA	https://www.sap.com/germany/products/erp/s4hana.html
PyCelonis	https://celonis.github.io/pycelonis/2.0.1/
pandas	https://pandas.pydata.org/
numpy	https://numpy.org/
anytree	https://anytree.readthedocs.io/en/latest/
matplotlib	https://matplotlib.org/
Git	https://git-scm.com/
Graphviz	https://graphviz.org/
Visual Studio Code	https://code.visualstudio.com/
MGWS Wiki	https://wiki.manroland-web.com/
MGWS Gitea	https://git.manroland-web.com/
MGWS Celonis EMS	https://manroland.eu-5.celonis.cloud/
Composite pattern	https://en.wikipedia.org/wiki/Composite_pattern
Python3 OOP	https://www.w3schools.com/python/python_classes.asp
Carbon	https://carbon.now.sh/
LEANX SAP Tabels	https://www.leanx.eu/en/sap/table/search