# Bachelorarbeit

## Transferieren eines Master Test Plans einer HGÜ Anlage in eine ALM Tool Umgebung

**Technische Hochschule Nürnberg Georg Simon Ohm**
**Fakultät Elektrotechnik Feinwerktechnik Informationstechnik**

Sommersemester 2020

TECHNISCHE HOCHSCHULE NÜRNBERG
GEORG SIMON OHM

SIEMENS
Ingenuity for life

| | |
|---|---|
| vorgelegt von: | Julian Leupold |
| Studienfach: | Elektro- und Informationstechnik |
| Matrikelnummer: | 3000866 |
| Erstprüfer: | Prof. Dr. Joerg Arndt |
| Zweitprüfer: | Prof. Dr. Helmut Herold |
| Firmenbetreuer: | Hr. Paul-Heinz Esters |
| Abgabedatum: | 31.07.2020 |

©2020

Dieses Werk einschließlich seiner Teile ist **urheberrechtlich geschützt**. Jede Verwertung außerhalb der engen Grenzen des Urheberrechtgesetzes ist ohne Zustimmung des Autors unzulässig und strafbar. Das gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikroverfilmungen sowie die Einspeicherung und Verarbeitung in elektronischen Systemen.

TECHNISCHE HOCHSCHULE NÜRNBERG
GEORG SIMON OHM

## Prüfungsrechtliche Erklärung der/des Studierenden

Angaben des bzw. der Studierenden:

Name: Leupold Vorname: Julian Matrikel-Nr.: 3000866

Fakultät: Elektro-, Feinwerk-, Informationstechnik Studiengang: Elektrotechnik und Informationstechnik

Semester: Sommersemester

**Titel der Abschlussarbeit:**

Transferieren eines Master Test Plans einer HGÜ Anlage in eine ALM Tool Umgebung

Ich versichere, dass ich die Arbeit selbständig verfasst, nicht anderweitig für Prüfungszwecke vorgelegt, alle benutzten Quellen und Hilfsmittel angegeben sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Obermässing, 22.07.2020 Julian Leupold

Ort, Datum, Unterschrift Studierende/Studierender

## Erklärung zur Veröffentlichung der vorstehend bezeichneten Abschlussarbeit

Die Entscheidung über die vollständige oder auszugsweise Veröffentlichung der Abschlussarbeit liegt grundsätzlich erst einmal allein in der Zuständigkeit der/des studentischen Verfasserin/Verfassers. Nach dem Urheberrechtsgesetz (UrhG) erwirbt die Verfasserin/der Verfasser einer Abschlussarbeit mit Anfertigung ihrer/seiner Arbeit das alleinige Urheberrecht und grundsätzlich auch die hieraus resultierenden Nutzungsrechte wie z.B. Erstveröffentlichung (§ 12 UrhG), Verbreitung (§ 17 UrhG), Vervielfältigung (§ 16 UrhG), Online-Nutzung usw., also alle Rechte, die die nicht-kommerzielle oder kommerzielle Verwertung betreffen.

Die Hochschule und deren Beschäftigte werden Abschlussarbeiten oder Teile davon nicht ohne Zustimmung der/des studentischen Verfasserin/Verfassers veröffentlichen, insbesondere nicht öffentlich zugänglich in die Bibliothek der Hochschule einstellen.

Hiermit ☒ genehmige ich, wenn und soweit keine entgegenstehenden Vereinbarungen mit Dritten getroffen worden sind,

☐ genehmige ich nicht,

dass die oben genannte Abschlussarbeit durch die Technische Hochschule Nürnberg Georg Simon Ohm, ggf. nach Ablauf einer mittels eines auf der Abschlussarbeit aufgebrachten Sperrvermerks kenntlich gemachten Sperrfrist

von 0 Jahren (0 - 5 Jahren ab Datum der Abgabe der Arbeit),

der Öffentlichkeit zugänglich gemacht wird. Im Falle der Genehmigung erfolgt diese unwiderruflich; hierzu wird der Abschlussarbeit ein Exemplar im digitalisierten PDF-Format auf einem Datenträger beigefügt. Bestimmungen der jeweils geltenden Studien- und Prüfungsordnung über Art und Umfang der im Rahmen der Arbeit abzugebenden Exemplare und Materialien werden hierdurch nicht berührt.

Obermässing, 22.07.2020 Julian Leupold

Ort, Datum, Unterschrift Studierende/Studierender

# Abstract

During the development of high-voltage direct current transmission systems, all test documents are included in a so-called Master Test Plan. These tests begin with factory tests of individual components and extend to acceptance tests at the customer's plant. This plan is currently based on an Excel-Sheet. It is processed by several departments over the course of the project. Since this process makes the plan extremely unclear and the multiple edits make it unstructured, it is to be transferred to an Application Lifecycle Management Tool. An associated workflow is also to be specified. In addition, an import and export mask is to be introduced. On top of this, the IEC 81346 and IEC 61355 standards have to be observed. After getting used to the online tool and the Master Test Plan, the first task to perform, was to define a structure for the converted plan. To now proceed, the variables and the input masks had to be created. Additionally some JavaScript scripts for simplyfing the use of the UI were implemented, based on a existing Polarion Extension. During the implementation a plugin has been developed as well. It executes a multiplication algorhytm based on the engineer standards mentioned above. The tests show, that Polarion ALM is a good way to manage plant data, but not for handling processing and multiplying many items. In addition some problems with the official manuals occured and the project is not working as it should. It would be advisable to process the data in an external tool before importing it to Polarion.

# Kurzübersicht

Bei der Entwicklung von Hochspannungs-Gleichstrom-Übertragungssystemen werden alle Prüfdokumente in einen so genannten Master-Test-Plan aufgenommen. Diese Tests beginnen mit Werksprüfungen einzelner Komponenten und reichen bis hin zu Abnahmetests auf der Anlage des Kunden. Dieser Plan basiert derzeit auf einer Excel-Tabelle. Er wird im Laufe der Projektabwicklung von mehreren Abteilungen bearbeitet. Da dieser Prozess den Plan extrem unübersichtlich und durch die mehrfachen Bearbeitungen unstrukturiert macht, soll er in ein Application Lifecycle Management Tool überführt werden. Ein zugehöriger Workflow muss ebenfalls implementiert werden. Darüber hinaus soll eine Import- und Exportmaske eingeführt werden. Zusätzlich sind die Normen IEC 81346 und IEC 61355 zu beachten. Nach der Einarbeitung in das Online-Tool und den Master-Testplan war die erste Aufgabe, eine Struktur für den konvertierten Plan zu entwickeln. Um nun fortzufahren, mussten die Variablen und die Eingabemasken erstellt werden. Zusätzlich wurden einige JavaScript-Skripte zur vereinfachten Nutzung der Benutzeroberfläche entwickelt, die auf einer bestehenden Polarion-Erweiterung basieren. Im Zuge der Implementierung wurde auch ein Plugin entwickelt. Es führt einen Multiplikationsalgorithmus aus, der auf den oben erwähnten Normen basiert. Nach dem Testen war klar, dass Polarion ALM eine gute Möglichkeit ist, Anlagendaten zu verwalten, aber nicht für die Verarbeitung und Multiplikation vieler Elemente geeignet ist. Darüber hinaus traten einige Probleme mit den offiziellen Handbüchern auf und das Projekt funktioniert nicht so, wie es sollte. Es ist ratsam, die Daten in einem externen Programm zu verarbeiten, bevor sie nach Polarion importiert werden.

# Glossar

**AC** Alternating Current

**ALM** Application Lifecycle Management

**API** application programming interface

**DC** Direct Current

**DCC** Document Classification Code

**DMS** Dokumenten Managment System

**HGÜ** Hochspannungsgleichstromübertragung

**HTML** Hypertext Markup Language

**ID** Identifikator

**IDE** integretated development environment

**LCC** line-commutated current-sourced converters

**MTP** Master Test Plan

**MTM** Master Test Matrix

**OTC** Object Type Catalogue

**PAM** Parameter Manager

**UI** Userinterface

**VSC** Voltage-Source Converter

**VTL** Velocity Template Language

**XML** Extensible Markup Language

# Abbildungsverzeichnis


2.1 Konzept des MTP . . . . . 4
2.2 Struktur . . . . . 6
2.3 Zusammensetzung des Objekttyps . . . . . 8
2.4 Struktur eines physikalischen Objekttyps . . . . . 9
2.5 Objektkennzeichen am Beispiel eines DC-Vorsteuerungsschrankes 10
2.6 Dokumentenkennzeichnung . . . . . 11
2.7 Bildung der Dokumentennummer . . . . . 12
2.8 Aspektschlüssel . . . . . 13
2.9 Beispiel des Userinterface am Typen „Test Case“ . . . . . 15
2.10 Beispiel des Fields XML-Schema . . . . . 16
2.11 Beispiel des Form-Konfiguartion XML-Schema . . . . . 17
2.12 Beispiel des Enumeration XML-Schema „Severity“ . . . . . 17

4.1 Projekt-Ablaufdiagramm . . . . . 26

5.1 Ablaufdiagramm des Skript-Worklfows . . . . . 34
5.2 Klassendiagramm der „WorkItemsMultiply“Erweiterung . . . . . 41
5.3 Ablaufdiagram der „doGet“-Methode . . . . . 43
5.4 Ablaufdiagramm der „getDependencies“-Methode . . . . . 44
5.5 Ablaufdiagramm der „Multiply“-Metode . . . . . 45
5.6 Ablaufdaigramm der Vervielfältigungsroutine, fallunabhängig . . 46
5.7 rendered.jsp . . . . . 47

8.1 Möglicher, weiterführender Workflow . . . . . 55

A.1 Schema zur Dokumentation für Leittechnik Objekte off-site . . . i
A.2 Schema zur Dokumentation für Leittechnik Objekte on-site . . . ii
A.3 Schema zur Dokumentation für nicht-Leittechnik Objekte off-site iii
A.4 Schema zur Dokumentation für nicht-Leittechnik Objekte on-site iv
A.5 Gesamte Benutzeroberfläche . . . . . v
A.6 Codebeispiel Velocity . . . . . vi

A.7 Testdocument Teil 1 . . . . . vii
A.8 Testdocument Teil 2 . . . . . viii
A.9 Preview eines Imports . . . . . ix
A.10 Fehlermeldung/Exceptionstack . . . . . x

# Inhaltsverzeichnis

**Prüfungsrechtliche Erklärung und Erklärung zur Veröffentlichung**

**Glossar**

**1 Einleitung** **1**

**2 Theoretische Grundlagen und Stand der Technik** **3**

2.1 Grundlagen HGÜ . . . . . 3

2.2 Grundlagen MTP . . . . . 4

2.2.1 Master Test Matrix . . . . . 5

2.3 Dokumentationsnormen IEC 61355, IEC 81346 und ihre Verwendung . . . . . 5

2.3.1 IEC 81346 . . . . . 5

2.3.2 IEC 61355 . . . . . 11

2.3.3 Dokumentennummer . . . . . 12

2.3.4 Dokumentendateiname . . . . . 13

2.4 Polarion . . . . . 14

2.4.1 Grundlegende Funktionalität . . . . . 15

2.4.2 Wiki Pages und Velocity . . . . . 18

2.4.3 API . . . . . 19

2.4.4 Extension FMC . . . . . 20

2.4.5 PAM . . . . . 21

2.5 Stand der Technik . . . . . 21

**3 Bedarfsanalyse** **23**

**4 Entwurf** **25**

4.1 Grundelegender Arbeitsblauf . . . . . 25

4.2 Gestaltung der Anwenderoberfläche . . . . . 26

4.3 Import- und Exportmasken . . . . . 27

4.4 Javascript Skripte auf der Basis FMC Work Item Save . . . . . 27
4.5 Multiplikationsverfahren . . . . . 28

**5 Implementierung** **29**
5.1 Workitem Testdocument . . . . . 29
5.2 JavaScript Skripte . . . . . 34
5.3 Multiplizierungsalgorithmus . . . . . 38
5.3.1 Wiki Page mit Velocity . . . . . 38
5.3.2 Plugin auf Basis des Beispiels „Servlet“ . . . . . 39
5.4 Importmasken . . . . . 48
5.5 Bedienungsanleitung . . . . . 48

**6 Tests** **49**

**7 Ergebnisse** **51**

**8 Zusammenfassung, Fazit und Ausblick** **53**
8.1 Zusammenfassung . . . . . 53
8.2 Fazit . . . . . 54
8.3 Ausblick . . . . . 55

**Literaturverzeichnis**

**A Anhang**
A.1 Dokumentationsschema . . . . . i
A.2 Polarion-Oberfläche . . . . . v
A.3 Velocity und API Code-Beispiel . . . . . vi
A.4 Interface Testdocument . . . . . vii
A.5 Importpreview . . . . . ix
A.6 Fehlermeldung bei „untoten“Workitems . . . . . x

# 1. Einleitung

Die immer weiter voranschreitende Digitalisierung bietet neben vielen Umstellungen auch viele Vorteile, wie zum Beispiel eine vollständige browserbasierte Application Lifecycle Management-Lösung [1] names Polarion, die es erlaubt Systeme definieren, erstellen, testen und verwalten zu können. Dieses System umfasst in dieser Arbeit eine HGÜ[1]-Anlage, deren Tests sowie der dokumentierten Ergebnisse dieser, nach Polarion ALM transferiert wurde. Vorangehende Pilotprojekte, die Implementierung der MTM[2], haben bereits gezeigt, wie vielversprechend und einfacher die Verwaltung des Produktzyklus über dieses browserbasierte Tool ist, als herkömmlich über per E-Mail versendete PDF Dateien oder riesige Excel-Tabellen. Das Ziel der Arbeit ist es, den MTP[3] in das Tool zu transferieren. Dabei ist es wichtig das Konzept zu verfolgen, auf dem der MTP basiert. Neben der Kenntnisse des Dokumentationssystem, definiert durch IEC 81346 und IEC 61355, und der Oberfläche von Polarion, bedarf es auch der Erstellung von Variablen und Eingabefeldern für das Userinterface. Dieses wurde abschließend noch formkonfiguriert und mit einigen wenigen JavaScript Funktionen benutzerfreundlicher gemacht. Aufgrund der Situation durch SARS-CoV-2 wurde diese Arbeit fast ausschließlich auf lokalen Instanzen und per Fernzugang von zu Hause aus erbracht.

[1] Hochspannungsgleichstromübertragung
[2] Master Test Matrix
[3] Master Test Plan

# 2. Theoretische Grundlagen und Stand der Technik

Die Erarbeitung der Arbeitsaufgabe macht ein intensives, dreiwöchiges Einarbeiten in folgende Thematiken notwendig.

## 2.1. Grundlagen HGÜ

Diese Arbeit beschäftigt sich grundlegend mit einer Anwendung, die eine HGÜ-Anlage darstellt und verwaltet. Deswegen wird hier kurz über die Grundlagen der Hochspannungsgleichstromübertragung gesprochen.
Folgender Absatz besteht aus Gedankenprotokollen mit Mitarbeitern der Firma Siemens und stichpunktartig aus [2, Kapitel 1-3]. Wie der Name schon sagt wird bei HGÜ, oder englisch HVDC (High Voltage Direct Current), elektrische Energie per Gleichstrom statt wie üblicherweise per Wechselspannung übertragen. Da die durch AC[1] enstehenden Blindleistungsverluste, zusätzlich zu den Wirkstromverlusten, durch Übertragungsleitungen ab gewissen Übertragungslängen, wie sie zum Beispiel bei Off-Shore Windparks anzutreffen sind, sehr groß und damit auch sehr teuer werden, lohnt es sich Energie über diese Strecken per DC[2] zu übertragen. Ab dem sogenannten break-even point überwiegt das Gleichstromübertragen in puncto Kosten der gewöhnlichen Wechselstromübertragung. Um diese Art der Energieübertragung zu bewerkstelligen bedarf es zweier Anlagen, eine am Einspeisepunkt der DC-Leitung und eine am Entnahmepunkt. Auf der einen Seite wird Spannung aus einem AC-Netz entnommen, gleichgerichtet und als DC übertragen. Auf der anderen Seite wird diese wieder wechselgerichtet und in ein AC-Netz gespeist. Je nachdem nach welcher Methode diese Umrichtung stattfindet spricht man von VSC[3] oder LCC[4].

[1] Alternating Current
[2] Direct Current
[3] Voltage-Source Converter
[4] line-commutated current-sourced converters

Diese Anlagen und deren Einrichtungen müssen in Betrieb genommen und getestet werden. Es gibt off-site-Tests auf dem Testgelände und on-site-Tests auf der Anlage.

## 2.2. Grundlagen MTP

„Der Master Test Plan beinhaltet alle Testdokumente, die im Rahmen einer Projektabwicklung durchgeführt werden. Dies beginnt bei Werksprüfungen (Routine und Typprüfungen) einzelner Komponenten bis hin zu den Abnahmetests auf der Anlage beim Kunden. Dieser Plan ist zurzeit noch auf Excel-Dokumenten basiert und deckt mehrere HGÜ Anwendungen ab. Dieser MTP wird von mehreren Abteilungen über die Projektlaufzeit bearbeitet.“ [3]
Diese Dokumente werden dort nur schriftlich geführt, nicht physisch hinterlegt. Er dient lediglich zur Übersicht aller Testdokumente. Das Konzept des MTP ist in untenstehender Abbildung zu sehen.

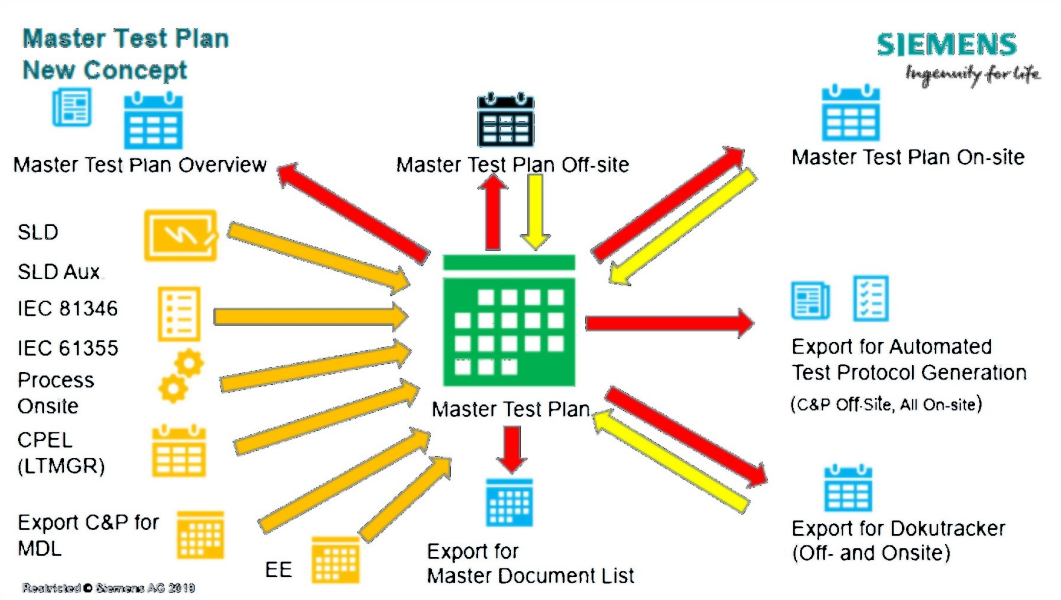


Abbildung 2.1.: Konzept des MTP, [4, Seite 2]

Er dient als Informationsursprung für weitere Dokumente und Arbeitsschritte im Laufe der Projektabwicklung. Durch Importe aus Listen und Diagrammen,

die auf Basis von Kundenspezifikationen erstellt werden, soll die Basis des MTP erzeugt werden. Einige dieser Dokumente sind auch bei Abnahme der Anlage oder bei Auswahl der Bauteile dem Kunden vorzulegen.

### 2.2.1. Master Test Matrix

Die MTM beschreibt verschiedene Testfälle und Testbedingungen, welchen die Anlage sowohl on- als auch off-site unterzogen wird. Diese Systemtests der Anlage werden auch im MTP dokumentiert.

## 2.3. Dokumentationsnormen IEC 61355, IEC 81346 und ihre Verwendung

Da genannte Normen äußerst umfangreich und tiefgreifend sind, aber nicht alle Informationen zum Bearbeiten des Themas benötigt werden, wird im folgenden nur auf die wichtigen Grundlagen eingegangen. Folgendes ist eine Zusammenfassung der relevanten Informatinen aus [5, Kapitel 1-5], [6, Kapitel 1-5], [7, Kapitel 1-4,6] und Gedankenprotokollen aus Mitarbeitergesprächen. Die Normen enthalten Aussagen und Regeln zur Klassifizierung von Objekten und Dokumentationen in Stationen für Verteilung und Übertragung elektrischer Energie. Gearbeitet wird zur Zeit noch mit dem Stand der Normen aus dem Jahr 2009 (IEC 81346-1/2) bzw. 2008 (IEC 61355).

### 2.3.1. IEC 81346

Die Kennzeichnung nach IEC 81346 bietet gegenüber früher genormten Kennzeichnungen weitergehende Möglichkeiten mit entsprechendem Rationalisierungspotential. In IEC 81346-1 kann nach Aspekten unterschieden werden, diese sind Produktaspekt, Ortsaspekt und Funktionsaspekt. Ein Aspekt beschreibt die Betrachtungsweise für das Objekt. Laut Siemens und einiger Kunden soll jedes Objekt nach dem Produktaspekt strukturiert werden. IEC 81346-1 strukturiert Objekte nach diesen Aspekten und lässt somit eine klare Anlagenzusammensetzung erkennen. IEC 81346-2 typisiert

Objekte in unterschiedliche Klassen und Unterklassen. Diese kommen auch bei der Strukturierung zum Einsatz. In HGÜ-Projekten wird die Dokumentenkennzeichnung nach obenstehenden Normen durchgeführt. Objektkennzeichen beschreibt im weiteren die Kennzeichnung nach IEC 81346-1 und IEC 81346-2, zusammengesetzt aus Referenzkennzeichen und Objekttyp.

### Referenzkennzeichen

Das Referenzkennzeichen besteht aus maximal acht Leveln, beginnend mit der Stationskennung. Danach folgt optional eine „tieferliegende“ Komponente. Die Strukturierung erfolgt meist von oben nach unten (top-down) oder von unten nach oben (bottom-up). Die Regeln erlauben im Prinzip einen Wechsel des Aspekts zwischen den Untergliederungsstufen einer Struktur, was jedoch in der Praxis selten zur Anwendung kommt.

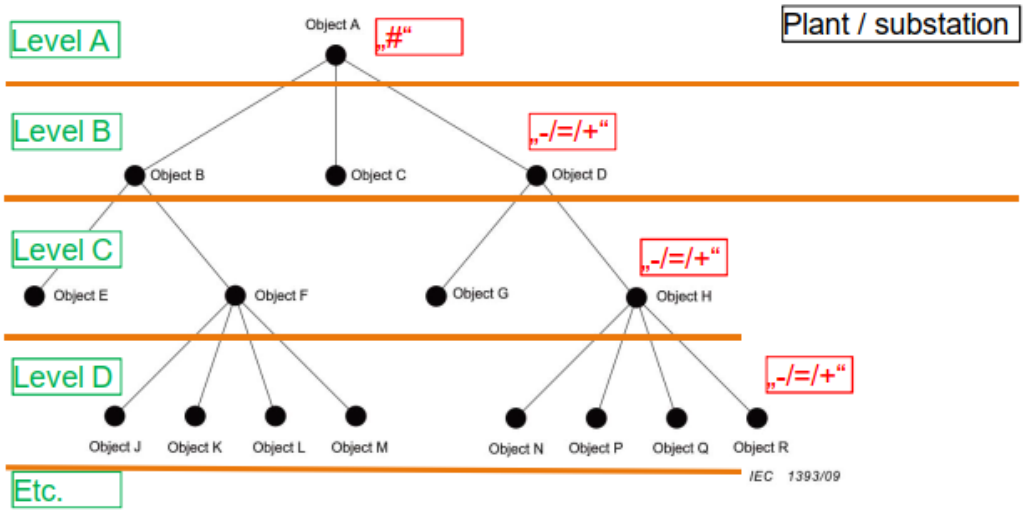


Abbildung 2.2.: Struktur, [8, Seite 10]

Level A bezeichnet immer die Station, Level B und weitere Level beschreiben je eine durch einen Kennbuchstaben zugeordnete Objektklasse. Die Kennbuchstaben sind in der Norm tabellarisch festgehalten. Einige Kennbuchstaben

haben zusätzliche Unterklassen, um die Objektklassen gemäß ihrer Aufgabe zu präzisieren. Sie werden durch zweistellige Kennbuchstaben definiert. Die Kennbuchstaben der Strukturebene B und C sowie darunterliegenden Ebenen unterscheiden sich grundsätzlich. Level B beschreibt Infrastrukturklassen, wie zum Beispiel „Umwandlung von Energie“ durch den Kennbuchstaben „T“, während Level C und alle darunter liegenden Level eine Klassifizierung nach Zweck oder Aufgabe beschreiben, zum Beispiel definiert „KF“ die Funktion „Verarbeitung von elektrischen und elektronischen Signalen“. Der Übergang in das nächst niedrigere Level wird durch ein Trennzeichen dargestellt . Bei einem Funktionssaspekt wäre der Seperator „=“, bei einem Ortsaspekt „+“ und bei einem Produktaspekt „-“. Erlaubt ist jedoch auch die Schreibweise „-A1B1C1“ oder „-A1.B1.C1“ statt „-A1-B1-C1“.
Normalerweise wird jedes Objekt bei Siemens angesichts des Produktaspektes strukturiert.
Jedes Referenzkennzeichen ist laut Norm einzigartig.

### Objekttyp

Der Objekttyp bedient sich den selben Objektklassen wie der Strukturlevel C des Referenzkennzeichens.
„Alle Objekttypen, die bei "Transmission Systems" verwendet werden, sind im OTC[5] enthalten." [7, Seite 16]
Intern wird das Kürzel OTC stellvertretend für die Objektklasse verwendet, ist vom OTC die Rede ist eigentlich die Objektklasse mit allen fünf Stellen gemeint. Alle Objekttypen sind wie folgt aufgebaut.

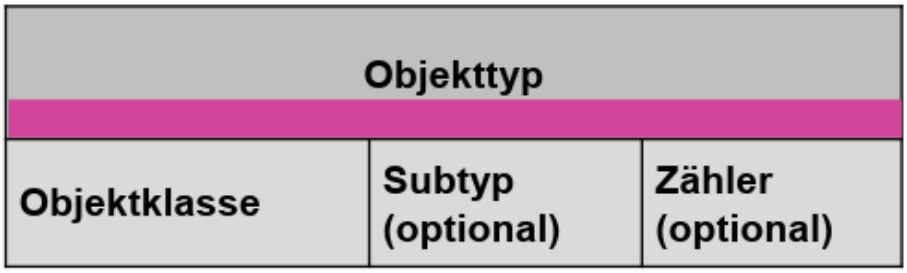


Abbildung 2.3.: Zusammensetzung des Objekttyps, [7, Seite 18]

Bei der Bildung des Objekttyps wird unterschieden in physikalische und nicht-physikalische Objekte.
Nicht-physikalische Objekte behandeln Dokumente, die keinem Produkt und keinem Ort eindeutig zugeordnet sind, nicht einem greifbaren Objekt/Produkt zugewiesen sind oder übergeordnet der Gesamtanlage dienen. Das sind zum Beispiel Studien oder Projektmanagementdokumente. Nicht-physikalische Objekte werden nur durch den Objekttypen gekennzeichnet. Sie haben kein Referenzkennzeichen. Nicht-physikalische Objektklassen werden durch fünf Zahlen gebildet, während physikalische Objektklassen aus zwei Buchstaben gefolgt von drei Zahlen bestehen.
Nicht-physikalische Objekte sind in aktueller Ausarbeitung noch durch Siemens definiert. Der Zahlencode der physikalischen Objekte wird ebenfalls durch Siemens definiert, die Norm gibt nur den Buchstabencode vor. Die Buchstaben entsprechen wie oben genannt dem Objektklassen-Code aus

[5] Object Type Catalogue

IEC8136-2.
Man erkennt in untenstehender Grafik deutlich, dass die Betrachtung des Objektes durch anhängen von drei Zahlen spezifiert wird.

| | |
|---|---|
| **KF312&E…** | **Objekttyp** für DC Vorortsteuerschrank |
| KF | Objektklasse aus dem RDS-System<br>“Verarbeitung von elektrischen und elektronischen Signalen” |
| KF300 | Systeme für Steuerung (und Schutz) |
| KF312 | DC Vorortsteuerschrank |

Abbildung 2.4.: Struktur eines physikalischen Objekttyps, [7, Seite 26]

## Objektkennzeichen

Nachfolgende Grafik veranschaulicht die Kombination eines physikalischen Objekttyps und des Referenzkennzeichens nach Produktaspekt zum Objektkennzeichen.

**KF312#XY11-AF04-KF01-UH02** DC Vorortsteuerung, Station 11, Schrank 2

**KF312** **Objekttyp** für DC Vorortsteuerschrank

**Referenzkennzeichen** nach Produktaspekt:

| | |
|---|---|
| #XY11 | Station A |
| #XY11-AF04 | Systeme für Bedienung, Beobachtung, Steuerung, Schutz |
| #XY11-AF04-KF01 | DC Vorortsteuerung, übergeordnet |
| #XY11-AF04-KF01-UH02 | DC Vorortsteuerung, Schrank 2 |

Abbildung 2.5.: Objektkennzeichen am Beispiel eines DC-Vorsteuerungsschrankes, [7, Seite 27]

### 2.3.2. IEC 61355

IEC 61355 erlaubt eine klar strukturierte Dokumentation zusammenhängend mit der eigenlichen Anlagenstruktur. In der unten stehenden Grafik sieht man die Zusammensetzung der Normen zum Dokumentenkennzeichen bzw. Dokumentenseitenkennzeichen.

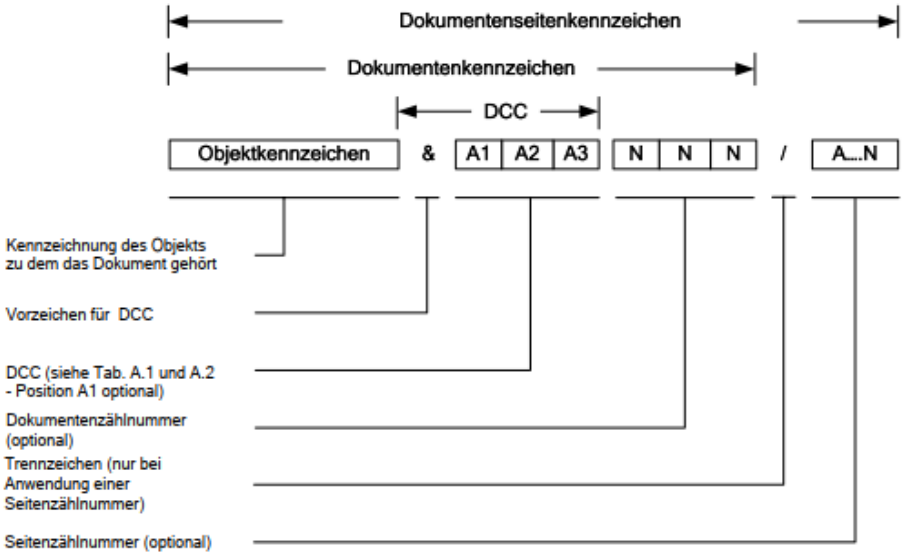


Abbildung 2.6.: Dokumentenkennzeichnung, [6, Seite 12]

Wichtigster Bestandteil dieser Norm ist der DCC[6].
Wie man in obiger Abbildung sieht, besteht der DCC aus einem Vorzeichen, einem folgenden Kennbuchstaben, der den technischen Bereich des Dokuments bezeichnet und einem zweistelligen Buchstabencode, der die Dokumentenartklasse beschreibt. Die optionale Dokumentenzählnummer spezifiziert das Dokument. Der MTP beinhaltet keinen DCC ohne Dokumentenzählnummer. So beschreibt `&EFS010` einen dem Bereich Elektrotechnik (einschließlich Steuerungs-, Informations- und Kommunikationstechnik) zugeordneten Stromlaufplan. Das Kürzel „FS“ beschreibt übergeordnet Schaltkreisdokumente, der

[6] Document Classification Code

Zahlencode „010“ spezifiziert auf Stromlaufplan. Optional können noch Seitenzahlen angegeben werden.
Alle physikalischen Objekte bei Siemens, auch „Equipment“ genannt, folgen einer in [9, Seite 47] festgelegten Logik, nach welcher über jedes Objekt, übergeordnet eingteilt in Leittechnik- und nicht Leittechnik-Objekt, on-site und off-site, immer die selben Dokumentationen geführt werden müssen. Das komplette relevante Schema ist in Anhang A.1 zu sehen.

### 2.3.3. Dokumentennummer

Die Anwendung beider Normen findet sich in der Bildung der Dokumentennummer wieder. Diese wird benötigt um alle Dokumente verschiedenster Art eindeutig zu kennzeichnen. Unter dieser Identifikationsummer werden diese später angelegt.

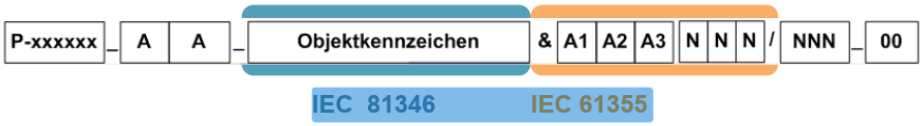


Abbildung 2.7.: Bildung der Dokumentennummer nach [7, Seite 8]

Hinzu kommen hierbei noch der Projektcode, die zwei folgenden Kennbuchstaben und die Revisionsnummer an letzter Stelle. Der erste Kennbuchstabe beschreibt die Dokumentationsart, ob intern, extern oder Angebot, der zweite, den Aspektschlüssel aus Norm 81346-1.

Der Aspektschlüsseltyp „C“, der ein Objekt klassifiziert, das keinem anderen Aspektschlüssel zugewiesen ist, wird hier noch ergänzt. Das sind meist übergeordnete Dokumente zu mehreren Komponenten, zum Beispiel Anlagenschaltpläne. Dieser ist nicht durch die Norm definiert, sondern intern.

Aspektschlüssel:

| | |
|---|---|
| P | Produktaspekt |
| L | Ortsaspekt (Location) |
| (F | Funktionsaspekt, nicht verwendet) |
| C | Objekttypenkatalog (OTC) |

Abbildung 2.8.: Aspektschlüssel nach [7, Seite 10]

In der Grafik nicht sichtbar und auch in Norm bzw. interner Dokumentation nicht erwähnt, gliedert sich vor der Revisionsnummer noch das Sprachenkürzel für die Sprache des Dokuments ein.
Ein komplettes Beispiel einer Dokumentennummer ist
`P-xxxxxx_EP_KF312#XY11-AF04-KF01-UH02&EFS010_EN_00` .

### 2.3.4. Dokumentendateiname

Das Ablegen der Dokumente in den Projektverzeichnissen erfolgt durch einen der Dokumentennummer ähnlichen Dateinamen. Dieser wird aus der Dokumentennummer erzeugt, indem alle Trennzeichen durch „_“ ersetzt werden. „-“, „+“ und „=“ entfallen. Dadurch wird das Referenzkennzeichen als Block zusammengefasst. Die einzige Ausnahme besteht darin, dass das „-“ nach der Stationskennung zu einem „_“ umgewandelt wird.
So ergibt sich
`P-xxxxx_EP_KF312_XY11_AF04KF01UH02_EFS010_EN_00` .

## 2.4. Polarion

Wie einleitend genannt, handelt es sich bei Polarion um ein browserbasiertes Tool, das Anforderungs-, Test- und Application-Lifecycle-Management, kurz ALM, liefert. [1, vergleiche]
„Polarion wird von globalen Unternehmen in einer Vielzahl von Branchen eingesetzt, wie zum Beispiel im Automobilbau, in der Medizintechnik und in der Luft- und Raumfahrt. Kunden erzielen mit Polarion die für die Herstellung Ihrer Produkte nötige Agilität, Traceability und Compliance. [...] Mehr als 2,5 Millionen Anwender weltweit vertrauen deswegen auf Polarion um die Zusammenarbeit in ihren Unternehmen voranzutreiben, ALM und Product Lifecycle Management (PLM) zu verknüpfen und um ihre hochwertigen Produkte auf den Markt zu bringen.“ [1]
Polarion läuft zentral auf einem Server, auf die der User per WEB-Browser zugreifen kann.
Die wichtigsten Funktionen, die zur Implementierung dieser Arbeit dienen, werden folgend genannt.

### 2.4.1. Grundlegende Funktionalität

In Polarion ist es möglich sogenannte Workitems einzupflegen, die an sich ein Objekt beschreiben. Dieses Objekt kann verschiedene Typen haben, dabei gibt es bereits fertige Standardtypen oder eigen erstellte „Custom-Types“. Das Userinterface eines jeden Workitem-Types lässt sich durch Erstellen neuer Felder bzw. das Umplatzieren dieser beliebig anpassen. Die in der Grafik erkennbare Kennung „workitem-502“ ist eine von Polarion selbst erzeugte ID, um jedes angelegte Workitem eindeutig zu kennzeichnen. Ein Ausschnitt der ganzen Benutzeroberfläche von Polarion ist im Anhang A.2 zu sehen. Darauf sieht man, zusätzlich zur Grafik unten, noch das Fenster, in dem alle Workitems eines Typen aufgelistet werden. Diese Auflistung kann durch Abfragen verschiedener Parameter (Bilden von „Queries“) strukturiert werden. Im Prinzip werden durch eine Query alle Workitems gefiltert und nach bestimmten Kriterien, zum Beispiel dem Inhalt eines Feldes, gelistet. Die Polarion-Query-Sprache ist grundsätzlich ähnlich zur Apache Lucene Syntax. [10]

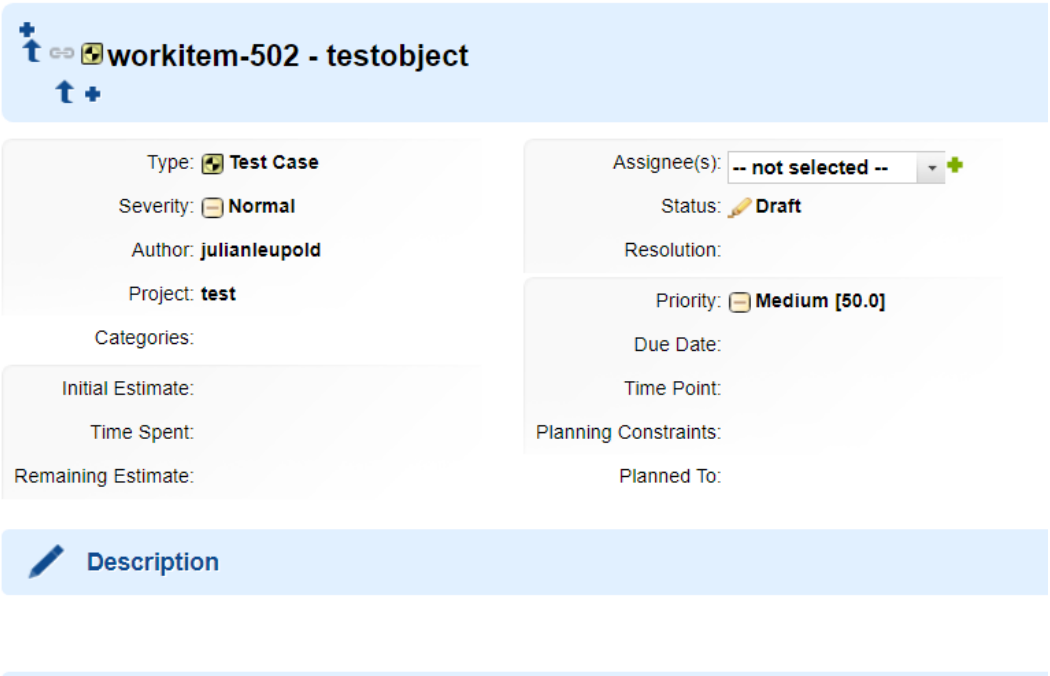


Abbildung 2.9.: Beispiel des Userinterface am Typen „Test Case“

Einige Felder des UI[7] sind bereits vordefiniert, meist werden sie vom System ausgefüllt, wie zum Beispiel das Feld `Author` , und standardmäßig, wie in Abbildung 2.9 zu sehen, positioniert. Andere, sogenannte „Custom-Fields“ lassen sich per XML[8]-Datei anlegen und in einer auch als XML vorliegenden „Form-Configuartion“ im Interface anordnen. Es gibt viele unterschiedliche Custom-Field-Types, die häufigsten sind jedoch Enumeration und String.

```
<fields>
  <field description="Workitem Type" id="type" name="Type" type="enum:type"/
      >
  <field id="severity" name="Severity" type="enum:severity"/>
  <field id="author" name="Author" required="true" type="string"/>
  <field id="project" name="Project" type="string"/>
  <field id="categories" name="Categories" type="string"/>
<\fields>
```

Abbildung 2.10.: Beispiel des Fields XML-Schema

Jede Felddefinition wird mit `<field` begonnen und endet mit `/>` . `description` bezeichnet den in einer Pop-Up Box sichtbaren Text, wenn die Maus im UI über den Anzeigenamen des Feldes bewegt wird. `name` beschreibt den Anzeigenamen des Feldes, „id“ den ID[9] des Feldes und `type` den Feldtypen. Mit `required="true"` wird festgelegt, welche Felder beim Anlegen eines einzelnen Items ausgefüllt sein müssen. Mit `multi="true"` ist es möglich bei Feldern vom Typen `enum` mehrere Optionen auszuwählen. Der Block der einzelnen Felder wird mit `<fields>` begonnen und mit `<\fields>` beendet.

[7] Userinterface
[8] Extensible Markup Language
[9] Identifikator

```
<!-- Horizontal layout element, adds components on horizontal row. Each
    component is in a new column. -->
    <horizontal>
        <!-- Vertical layout element adds components into one vertical
            column. Each component is in a new row. -->
        <vertical>
            <section>
                <field id="type" readOnly ="true" />
                <field id="severity" />
                <field id="author" />
                <field id="project" />
                <field id="categories" />
            </section>
      </vertical>
    </horizontal>
```

Abbildung 2.11.: Beispiel des Form-Konfiguartion XML-Schema

Wie im Beispielcode zu sehen beginnt `<horizontal>` eine Reihe, `<vertical>` eine Spalte und `<section>` ein Element dieser Spalte. Wird ein Feld ohne Section definiert, so erscheint es als blauer Balken, wie `Description` in Grafik 2.9. Es gibt die Möglichkeit Felder mit `readOnly="true"` nur lesbar zu setzen. Wenn ein Custom-Field als „enum“ definiert wird, muss in einem weiteren XML-Schema die Enumeration definiert werden. Enumerationen werden dann gewählt, wenn der Benutzer aus einem dropdown Listenfeld vorgegebene Werte auswählen soll.

```
<enumeration>
  <option id="blocker" name="Blocker" description="" sortOrder="1"/>
  <option id="critical" name="Critical" description="" sortOrder="2"/>
  <option id="major" name="Major" description="" sortOrder="3"/>
  <option id="normal" name="Normal" description="" sortOrder="4" default="
      true"/>
  <option id="minor" name="Minor" description="" sortOrder="5"/>
  <option id="trivial" name="Trivial" description="" sortOrder="6"/>
</enumeration>
```

Abbildung 2.12.: Beispiel des Enumeration XML-Schema „Severity“

Jeder auswählbare Wert wird hier als `option` aufgeführt. Über den `id` kann später darauf zugegriffen werden, `name` gibt den Anzeigenamen an und `description` , wie auch oben, den Text in der Pop-Up Box, die angezeigt wird, sobald die Maus über eine Option bewegt wird. `sortOrder` beschreibt die Reihenfolge in der die Optionen in der Klappliste angezeigt werden.

Zusätzlich zu oben genannten Features verfügt Polarion über eine Import-Funktion aus Excel-Workbooks und den Excel-Roundtrip-Change Import, bei dem bestehende Datensätze zuerst aus Polarion exportiert werden, bearbeitet und wieder importiert werden. Exportieren ist in fast alle gängigen Anzeigeformate möglich.
Verschiedene Benutzerrechtegruppen lassen sich auch erstellen.
Außerdem lassen sich in Polarion unterschiedlche Workitems miteinander verlinken, was es möglich macht Baumstrukturen zu erstellen, oder einen reinen Dokumentennamen mit dem tatsächlichen Dokument in Beziehung zu stellen.

### 2.4.2. Wiki Pages und Velocity

Wiki-Pages sind editierbare Dokumente, die zum Anzeigen von Daten, Objekten und Informationen dienen sollen. Diese Dokumente stehen für jedes Projekt in einem übergeordneten Verzeichnis zur Verfügung. Sie sind Workitemtyp-unabhängig. Ein einfaches Beispiel hierfür wäre die Anzeige aller Workitems, die mit einem bestimmten Flag, zum Beispiel „eilig“ versehen sind. Diese „Pages“ können mit der Java-basierten Template Engine „Velocity“ ([11][vergleiche]) editiert werden. Die verwendete Sprache, genannt VTL[10] lässt sich komplett in den geschriebenen Text integrieren und bietet einige, wenn auch eingeschränkte, Funktionalitäten. Diese Sprache soll „die einfachste, unkomplizierteste und sauberste Möglichkeit bieten, dynamische Inhalte in eine Webseite zu integrieren.“ [11].
Zusätzlich dazu stehen einige Templates aus Polarion zur Verfügung, die die Anzeige von Datensätzen vereinfachen oder einfach nur die Möglichkeit bieten

[10] Velocity Template Language

Bilder in die Wiki-Page zu integrieren.
Ein Zugriff auf die Polarion API[11] ist über bereits instanziierte Objekte der Services möglich, die einen Einstiegspunkt in die API ermöglichen.
Polarion arbeitet nicht mit der aktuellsten Version der VTL, sondern mit Version 1.4. [12, vergleiche Seite 8]
Ein kurzes Code-Beispiel unter Verwendung der API befindet sich in Anhang A.3.

### 2.4.3. API

Polarion verfügt über eine Java API, mit der es möglich ist Workitems zu ändern, zu lesen und zu erstellen, Benutzer zu verwalten, nach Workitems zu filtern oder Projekte aufzulisten, nur um ein paar Funktionen zu nennen. Zur Verfügung stehen verschiedene Services, mit unterschiedlichen Funktionen, mit denen der Zugriff auf den Polarion Server ermöglicht wird. ([13][vergleiche])
Ein paar der Wichtigsten sind:

#### ISecurityService

„The entry point for authentication and authorization tasks. The SecurityService manages the users, roles, relationships and permissions.“ [13, Seite 3]

#### ITrackerService

„The main entry point for tracker-related functions. They are divided by domains into individual managers that provide the following: search, read, creation, modification and the linking of Work Items and their properties such as; work records, attachments, comments and time points.“ [13, Seite 3]

[11] application programming interface

#### ITransactionService

„This service provides a way to persist changes to the repository with several mechanisms like wrapping exceptions and the ability to roll them back in case of error or performing commit action.“ [13, Seite 3]

#### IDataService

„It provides operations with data objects that can be persisted like searching, saving, resolving and viewing their history via revisions.“ [13, Seite 3]

### 2.4.4. Extension FMC

Informationen über die Funktionsweise des Plugins, wurden dessen öffentlich einsehbaren Quellcodes bzw. der Funktionsanleitung entnommen. ([14])
Die zur Verfügung gestellten Server haben bereits eine Extension vorinstalliert, die es möglich macht eigen verfasste Skripte in JavaScript in den Polarion-Wokflow zu integrieren. In diesen kann eingeschränkt auf die API zugegriffen werden, es stehen nur Instanzen des aktuellen Workitems, des „TrackerService“ und eines Loggers zur Vefügung.
Das Plugin überschreibt die von Polarion aufgerufene „invoke“-Funktion, falls der „save“-Parameter gesetzt ist und lässt somit nach dem Speichern eines Workitems, das die „invoke“-Funktion mit dem gesetzten „save“-Parameter aufruft, die Ausführung von eigenem Code zu. Sobald die im Code verfügbare Variable „returnvalue“ über einen Inhalt verfügt, wird dem User eine Error-Message angezeigt, die diesen Inhalt darstellt. Das Speichern wird somit verhindert.
Ein einfaches Beispiel wäre zum Beispiel eine Überprüfung auf das Einhalten eines Prozentsatzes. Eine Kombination aus drei Feldern soll einen Prozentsatz von 100% ergeben, bei einer Ungleichheit zu 100% soll ein Speichern nicht möglich sein.

### 2.4.5. PAM

Folgende Informationen stammen aus einer Konversation mit Hr. Paul-Heinz Esters. PAM[12] ist ein Tool, um Parameter eines Parameter Daten Modells zu editieren und zu verwalten. Es basiert auf einem Git Versions Control System. Es kommt bisher zur Parameterierung der Anlagensoftware, bei der Erstellung des 3D-Anlagen Modells und bei der Parametrierung, nach IEC 61850, der Kommunikation auf der Anlage zum Einsatz. In PAM sollen zukünftig alle anlagenbezogenen Daten, von Nenndaten bis zu den Referenzkennzeichen der Bauteile gesammelt werden. Somit wird PAM zum zentralen Daten-Backbone für HGÜ Anlagen erweitert.

## 2.5. Stand der Technik

Aktuell werden die Datensätze des MTP über eine Excel-Tabelle verwaltet. Auf ein bestehendes Template wird über verschiedene Informationszugänge von Hand der schlussendliche Datensatz aufgebaut. Dieser Ansatz ist zwar funktional, aber grundsätzlich unhandlich und ohne einsichtlichen Userworkflow. Zusätzlich ist kein ausreichender Schutz vor Datenverlust bei Mehrfachzugriff möglich. Zu Arbeiten mit Polarion gibt es aktuell ein Pilotprojekt, das die MTM behandelt, die ähnlich wie der MTP über Excel verwaltet wurde.

[12] Parameter Manager

# 3. Bedarfsanalyse

Das Ziel der Arbeit ist das Transferieren eines MTP einer HGÜ-Anlage in die Umgebung von Polarion.
Die Oberfläche soll der Excel-Tabelle ähneln und für den Anwender leicht zu bedienen sein. Folgende Punkte sind zu berücksichtigen:

(i) Hinterlegung eines Workflows im Tool (Zugriffsrechte auf verschiedene Phasen und Komponenten je nach Projektrolle)
(ii) Import- und Exportfunktionen aus dieser Datenbasis
(iii) Berücksichtigung der Dokumentationsvorgaben (Kapitel 2.3)
(iv) Definition der erforderlichen Variablen, Erstellen von Eingabemasken (UI)
(v) Integration eines Trackings über den Dokumentationsstand

Punkt (i) ist zu Beginn gedacht, wird jedoch während der Bearbeitung verworfen bzw. verschoben. Wie in Kapitel 2.4 genannt, gehört eine Import-/Exportfunktion bereits zum Funktionsumfang von Polarion. Jedoch ist es notwendig Importmasken zu definieren. Während der Bearbeitung kommt die Idee auf, einen auf den Regeln zum Dokumentationsschema in Kapitel 2.3.2 basierenden Workitem-Multiplikationsalgorithmus zu erstellen.

# 4. Entwurf

Bevor erste Ansätze enstehen, werden intensive Einarbeitungsgespräche mit Betreuer Paul-Heinz Esters geführt, die dem Zweck dienten einen ersten groben Überblick der Thematiken zu erhalten. Daraufhin erfolgt wie einleitend genannt eine Einarbeitungs- und Lernphase, durch mehrere Dokumente, Lernvideos und einem Admin-Key Training zu Polarion. Zusätzlich dazu werden Gespräche mit dem Kollegen Andoni Sainz Lopez geführt, die über die API und die Erweiterung aus Kapitel 2.4.4 informieren.

## 4.1. Grundelegender Arbeitsblauf

Zu Beginn soll der Ablauf des Projekts festgelegt werden. Dazu gehörte das Erstellen eines übergeordneten Workflow-Diagramms. Der entgültige Arbeitsablauf steht erst zu Ende des Projekts fest, wird jedoch hier aufgeführt. Aufgrund der COVID-19 Pandemie laufen viele Unternehmensabläufe erschwert ab, die Kommunikation ist eingeschränkt und Tagungen von Ausschüssen zur Besprechung von internen Standards finden nicht oder verspätet statt. Das führt zu Verspätungen von Datenstandards, die zur Festlegung des Arbeitsablaufs nötig sind. Zudem ergibt sich die in Kapitel 3 genannte Idee zur „Datensatz-Multiplikation“ erst im Laufe der Bearbeitung. Wie in 2.2.1 erläutert erfasst, der MTP abschließend auch die Testdokumente der Systemtests der MTM. Da dieses Projekt, wie in Kapitel 2.5 ausgeführt, allerdings noch nicht vollständig abgeschlossen ist, wird in der Implementierung nicht darauf geachtet.
Außerdem sind zu gewissen Zeitpunkten, hier nicht aufgeführte, Exporte in das DMS[1] vorgesehen.

[1] Dokumenten Managment System

Im Folgenden ist der Ablauf geschildert. Die Funktonsweise bzw. der Zweck der Skripte wird später erläutert.

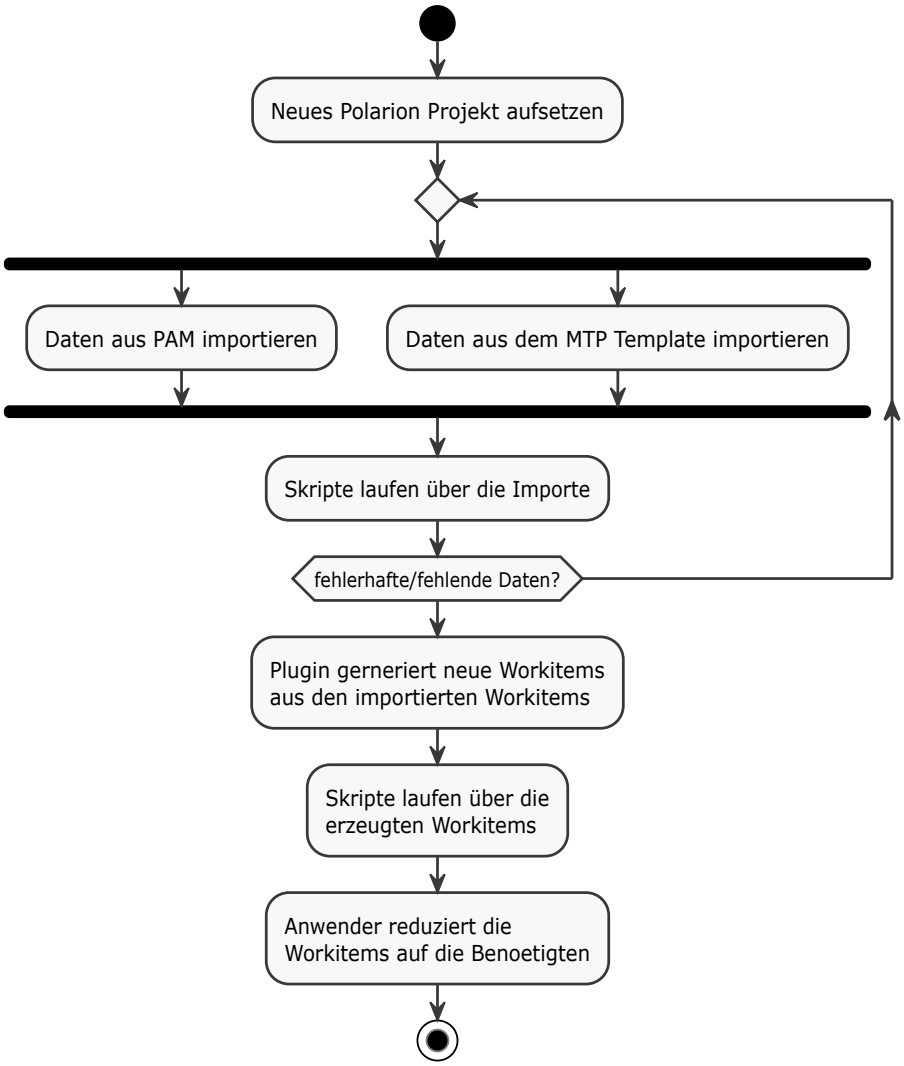


Abbildung 4.1.: Projekt-Ablaufdiagramm

## 4.2. Gestaltung der Anwenderoberfläche

Die Anwenderoberfläche in Polarion soll der Exceloberfläche so ähnlich wie möglich sein, um dem Benutzer die Umstellung auf das neue Portal so einfach wie möglich zu machen.

Grundsätzlich bedarf es einem neuen Workitem Typ. Die benötigten „Custom Fields“ werden auf Grundlage des Excel-MTP geplant, für Felder, die mit vordefinierten Inhalten gefüllt sind, werden Enumerationen vorgesehen. Allgemein soll jedes Feld der Excel-Vorlage in die Polarion-Version übernommen werden. Je mehr Felder als Enumerationen konfiguriert werden können, desto einfacher ist es fehlerhafte Eingaben durch den Anwender zu vermeiden. Da viele Felder, vorallem projektspezifische, wie zum Beispiel die Stationskennung, nicht vorhersehbar sind, werden diese als Typ „string“ konfiguriert. Eine Form-Konfiguartion, nach Abbildung 2.11, wird zuerst theoretisch überlegt und auf Papier erstellt.

## 4.3. Import- und Exportmasken

Da das schlussendliche Projekt aus einer Mehrzahl an Datenquellen zusammengesetzt wird, muss feststehen, welche Informationen aus dem jeweiligen Import benötigt werden.
Wie in Abbildung 4.1 zu sehen ist, gibt es aktuell zwei Datenquellen für das Aufsetzen eines jeden Projektes. Aus der Excel-Vorlage des MTP werden alle Items importiert, die den Aspect-Key Typ „C“ haben. Diese sind größtenteils für jedes Projekt gleich, sodass nur kleine Änderungen vorgenommen werden müssen.
Aspektschlüsseltypen „P“ und „L“ werden aus der in Kapitel 2.4.5 erwähnten PAM-Datenbank importiert. Typ „F“ wird aktuell nicht geführt. Dadurch wird jedes physische Objekt ohne DCC in den MTP aufgenommen.

## 4.4. Javascript Skripte auf der Basis FMC Work Item Save

Um dem User das Bedienen der Oberfläche zu vereinfachen wird mit Skripten des FMC-Einschubprogramms (Plugin), Kapitel 2.4.4, gearbeitet. Diese „bauen“ zum Beispiel die, aus Kapitel 2.3.3 und 2.3.4, Dokumentennummern und Dokumentendateinamen aus den einzelnen Bestandteilen zusammen. Es sollen so wenig Daten wie möglich importiert werden müssen, um die Importmasken

bzw. die Exportmasken aus PAM so klein wie möglich zu halten. Deshalb füllen die Skripte das Workitem mit Daten, die aus dem kleinst-möglichen Import gewonnen werden können.

## 4.5. Multiplikationsverfahren

Dieser Teil orientiert sich an 2.3.2, Absatz 4. Da die Datensätze der physikalsichen Objekte (Aspektschlüsseltyp P und L) der Anlage durch den Import aus Kapitel 4.3 ohne DCC vorhanden sind, werden diese durch Vervielfältigung und Anhängen eines DCC zu vollwertigen Dokumenten-Datensätzen. Zusätzlich dazu gab es noch weitere Ideen, um die Datensätze zudem aufzubereiten. Diese wurden aufgrund von Bedenken im Bezug auf die Leistungsfähigkeit des Systems verworfen, werden jedoch im weiteren Verlauf kurz erwähnt.
Für den Multiplikationsalgorithmus gibt es zwei Konzepte, eines in Velocity, eingebettet in einer Wiki-Page, das andere als Erweiterung zu Polarion, auf Basis eines bereits vorhandenen Beispiels. Das erste Konzept erweist sich als unbrauchbar und wird verworfen. Der theoretischer Ansatz des grundlegenden Algorithmus bzw. Aufbaus der Routine der beiden ist identisch.
Grundsätzlich soll es möglich sein Informationen durch den Anwender zu übergeben.

# 5. Implementierung

Sämtlicher XML/Code sowie weitere Dateien sind bei Veröffentlichung der Arbeit unter folgendem Link abrufbar: `https://github.com/syxlox/testdocument.git`.

## 5.1. Workitem Testdocument

Die Implementierung der neuen Benutzeroberfläche, um den MTP nach Polarion zu überführen, setzt die Erstellung eines eigenen Workitemtypen und einiger XML-Dateien voraus.
Der neue Workitemtyp „testdocument“ wurde in Polarion mit allen vom System vorgegebenen Feldern und Werten angelegt. Statt wie ursprünglich geplant, wird nicht jedes Feld des Excel-Plans in Polarion übernommen, sondern nur die wichtigsten. Der Excel-Plan ist über einige Jahre entstanden und während einige Felder nicht mehr benötigt werden, sind andere durch Funktionalitäten aus Polarion abgelöst. Diese werden bewusst nicht übernommen. Implementiert wurden insgesamt 60 „Custom-Fields“, von denen vier allein der späteren Formkonfiguration dienen.
Folgende Felder wurden erstellt, ausgenommen den oben genannten vier. Zu allen, in den Augen des Verfassers nicht offensichtlich ersichtlichen Feldern, wird ein kurzer Absatz gesagt. Die Felder sind in sinnhaften Blöcken angeordnet.

- TestingSite, type=enum
- DMS Identifier, type=string
- VSC, type=enum
- LCC, type=enum
- Object Type, type=enum
- Serialnumber, type=string

Um später durch Bilden von Queries zwischen, in Kapitel 2.1 genannten, off-site und on-site Dokumenten unterscheiden zu können, wird hier das Feld „TestingSite“ angelegt. Der „DMS-Identifier“ dient dazu, um Objekte bei späteren Exporten in das DMS eindeutig zu kennzeichnen. Unter „VSC“ und „LCC“ wird festgehalten, ob ein Dokument für den jeweiligen Anlagentyp optional oder erforderlich ist. Bei Kennzeichnung mit optional, kann später durch den Benutzer entschieden werden, ob das Dokument entfernt wird oder nicht. Der „Object Type“ beschreibt, ob das Objekt physikalisch oder nicht-physikalisch ist.

- Documentnumber, type=string
- Documentfilename, type=string
- Document Type, type=enum
- Aspect Key, type=enum
- OTC Code, type=string
- Subtype, type=string
- Station number, type=string
- Level B, type=string
- Level C, type=string
- Level D, type=string
- Level E, type=string
- Level F, type=string
- Level G, type=string
- Level H, type=string
- Code Letter, type=enum
- DCC Class, type=enum
- Language, type=enum
- Page Counter, type=string
- Revision, type=string
- OTC.Subtype, type=enum

Dieser Block implementiert die in Kapitel 2.3 genannten, normgerechten Bausteine der Dokumentennummer bzw. des Dokumentendateinamens. Die Projekt-ID wird übergeordnet für alle Workitems gleich angelegt, muss also hier nicht extra erzeugt werden. Für spätere Exporte wird das Feld „OTC.Subtype“ angelegt, dass den OTC und seinen optionalen Subtypen in einen String zusammenfasst.

- Station number, L, type=string
- Level B, L, type=string
- Level C, L, type=string
- Level D, L, type=string
- Level E, L, type=string
- Level F, L, type=string
- Level G, L, type=string
- Level H, L, type=string

Die durch diesen Block erstellten Level des Referenzkennzeichens, dienen dazu, falls ein Objekt bereits durch den Aspekt „P“ beschrieben wird, hier zusätzlich optionale Informationen über die Lokation des Objekts angeben zu können. Das ist besonders auf der Baustelle hilfreich, da so das physikalische Gegenstück des Dokuments schneller gefunden werden kann. Sollte der Aspekt des Dokuments bereits „L“ sein, entfallen diese Informationen.

- Documenttitle, type=string
- DCC description, type=string

Diese zwei Felder werden durch die in Kapitel 4.4 erwähnte Methodik mit den Beschreibungen der jeweiligen OTC bzw. DCC befüllt. Die Beschreibung des OTC dient gleichzeitig als Dokumententitel.

- Responsible Department, type=enum
- Responsible SubPM, type=enum

Die Abteilungsbezeichnung erfolgt in Feld „Responsible Department“. Feld „Responsible SubPM“ (Sub-Project Manager) beschreibt den Teilprojektleiter.

- issued for information, type=enum
- issued for approval, type=enum
- issued for design, type=enum
- issued for manufacture, type=enum
- issued for construction, type=enum
- as built, type=enum

Die Felder dieses Blocks beschreiben eine übergeordnete Thematik des Dokuments. Zum Beispiel werden Prüfbescheinigungen `(&* QC04*)` immer als „issued for information“ gekennzeichnet, während bei Inbetriebsetzungsanleitungen `(&* DC03*)` „issued for approval“ vermerkt wird. Dadurch können später die richtigen Suchanfragen gebildet bzw. dem Kunden, die für ihn relevanten Dokumente, zur Verfügung gestellt werden.

- Contact Person at Siemens, type=string
- Manufacturer, type=string
- Contact Person at Manufacturer, type=string
- Work Progress, type=enum
- Originator, type=string
- Due Date, type=date
- Review Status, type=enum
- Reviewer, type=string
- Due Date, type=date
- Approval Status, type=enum
- Approver, type=string
- Due Date, type=date

Durch diese Custom-Fields sind größtenteils Ansprechpartner bzw. allgemein Personen von Interesse im Bezug auf dieses Objekt vermerkt. Das „Work Progess“ Enumeration-Feld erfüllt die Aufgabe den Fortschritt zu überwachen, indem sich mit eigen erstellten Icons verschiedene Fortschritte eintragen lassen (0%,25%,50%,75%,100%).

Zu fast allen hier als Typ „enum“ aufgeführten Feldern wurde eine gleichnamige Enumeration erstellt, außer für die jenigen, die nur mit „ja“ oder „nein“ anzuwählen waren. Für diese wurde ein einziges, immer wieder verwendetes,

XML-Schema „enum:yon“ erstellt. Alle Enumerationen folgen dem in Kapitel 2.4.1 gezeigten XML-Schema. Die Enumeration des DCC und OTC wurden aus den jeweiligen Listen entnommen, nach Schema aufbereitet und importiert, so dass jeder Code bzw. seine Beschreibung (zum Beispiel DC030 = Inbetriebsetzungsanleitung) in Polarion enthalten ist. Der `id` der Option ist immer der jeweilige Code bestehenden aus Kleinbuchstaben und die Beschreibung des Codes der Inhalt der „description“ Variable. Jeder einzelne DCC bzw. OTC hat immer eine Beschreibung.
Die restlichen Enumerationen beinhalten alle grundsätzlich logische Optionen, die hier nicht gesondert aufgezählt werden.

### Form Konfiguartion

Die vier restlichen Custom-Fields wurden als Abschnittsüberschriften für die Form-Konfiguration benutzt. Sie erscheinen als „Überschriften“ der einzelnen Blöcke, zum Beispiel „doc_numbering“. Die Felder wurden möglichst übersichtlich und logisch angeordnet.
Das formkonfigurierte Interface zum Workitem „Testdocument“ ist ab Anhang A.4 zu sehen. Alle Abschnitte ab „Comments“ sind standardmäßig bei allen Workitem-Types aufgeführt und werden deshalb nicht mit abgebildet.

## 5.2. JavaScript Skripte

Insgesamt werden acht Skripte verfasst, die gemäß der „readme“-Datei der FMC-Erweiterung ([14]) in einem übergeordneten Skript „testdocument-pre-save.js“ nach dem, in der Grafik zu sehenden Ablaufdiagramm, strukturiert werden.

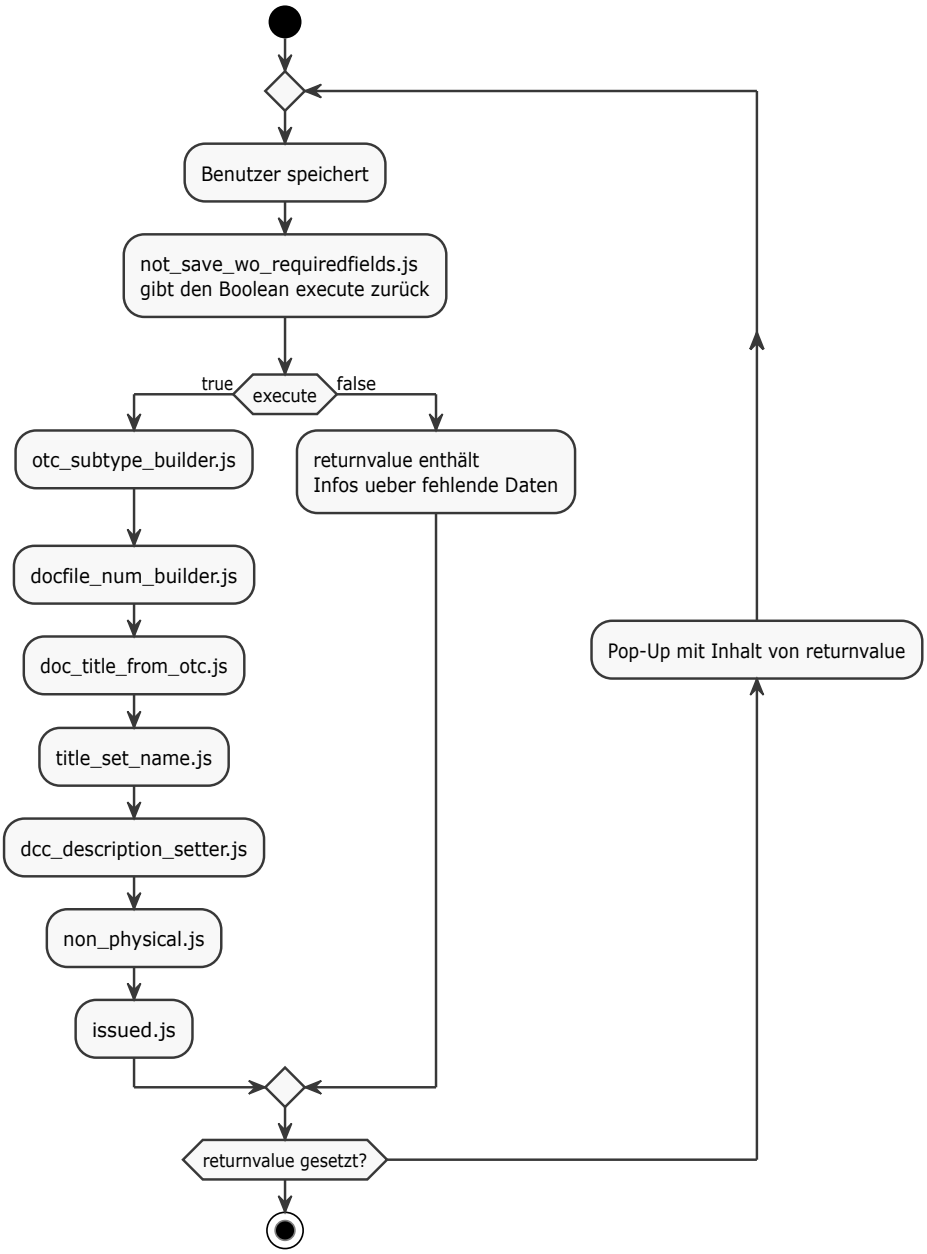


Abbildung 5.1.: Ablaufdiagramm des Skript-Worklfows

Nachfolgend wird zu jedem einzelnen Skript kurz dessen Grundfunktion erläutert.

### not_save_wo_requiredfields.js

Es ist in Polarion zwar möglich, bestimmte Felder als „required“ zu setzen, das heisst beim Anlegen eines Workitems müssen diese Felder immer gesetzt sein, allerdings ist diese Funktionalität leicht umgänglich, da zum Beispiel bei Importen nicht darauf geachtet wird.
Dieses Skript setzt eingehend den Standardwert der Dokumentensprache auf „EN“, sofern kein Wert eingetragen ist, da auch hier die Polarion-eigene Funktion, zur Standardwert-Vergabe, nicht funktioniert.
Danach wird geprüft, ob die für die restlichen Skripte erforderlichen Felder „OTC“, „Aspect Key“ und „Document Type“ gesetzt sind. Ist das nicht der Fall, wird der Boolean „execute“ auf „false“ gesetzt und somit die restlichen Skripte nicht ausgeführt. Zusätzlich wird die Variable „returnvalue“ mit den fehlenden Daten beschrieben und anschließend dem Benutzer anhand einer, durch die Erweiterung erzeugten, Pop-Up Nachricht dargestellt.

### otc_subtype_builder.js

Dieses Skript nimmt den Inhalt der als String importierten Felder „OTC“ und „Subtype“, und setzt daraus das Enum „OTC.Subtype“ zusammen. Dadurch wird auch überprüft, ob die eingegebenen Strings zusammengesetzt einen logischen Wert ergeben. Ist das nicht der Fall, versucht die Methode ([15]) das Feld zu setzen, wobei aber bei einem übergebenen Argument, dass nicht als „Enum-Option“ verfügbar ist, das Enum-Feld auf „null“ gesetzt wird. In diesem Fall wird „returnvalue“ mit einem Hinweis auf eine fehlerhafte Eingabe beschrieben.

### docfile_num_builder.js

Das Skript implementiert die in Kapitel 2.4.4 als Beispiel genannte Funktion, die Dokumentennummer, sowie den Dokumentendateinamen, aus ihren Bausteinen zusammenzusetzen. Für die Bausteine des Referenzkennzeichens

wird aus Performance-Gründen eine Schleife mit Abbruchbedingung verwendet. Da das Referenzkennzeichen niemals ein Hierachielevel auslässt, kann ab dem Nichtvorhandensein eines einzelnen Levels die Bildung des Referenzkennzeichens beendet werden.
Da nicht alle Bausteine String-Felder sind, kann auch nicht alles in einer Schleife realisiert werden. Um auf den Inhalt eines Enum-Feldes zuzugreifen, muss die Methode „getName()“ verwendet werden, welche bei String Feldern entfällt.
Das Ersetzen der Steuerzeichen, der zuerst generierten Dokumentennummer, nach Kapitel 5.2, erfolgt durch die JavaScript-Methode „replace“ [16] mit einem regulären Ausdruck als Argument. Da jedoch an einer Stelle diese Logik außer Kraft tritt, werden Dokumentennummer, sowie Dokumentendateiname in zwei Blöcke unterteilt und schlussendlich mit dem Trennzeichen „-“ bzw. „_“ zusammengesetzt.

### doc_title_from_otc.js

Der als Enumeration angelegte OTC führt zu jeder „Enum-Option“ die Bezeichnung als „decription“ der Enum-Option mit.
Durch die API-Methode „getProperty()“ mit dem Argument „description“ kann die Beschreibung der aktuell gesetzten Option abgefragt werden. Diese wird dann als Inhalt des Feldes „Documenttitle“ gesetzt und für spätere Zwecke in der gloablen Variable „doctitle“ abgespeichert.

### title_set_name.js

„title_set_name.js“ kombiniert die Bezeichnung des OTC, abgespeichert in „doctitle“, und die Dokumentennummer zu einem String und setzt diesen als Workitem-Titel. Falls die Dokumentennummer leer ist, wird nur die Bezeichnung des OTC als Titel gesetzt. Wenn im aktuellen Titel das Steuerwort „skip“ steht, wird kein neuer Titel gesetzt, sondern der aktuelle beibehalten.

### dcc_description_setter.js

Dieses Skript funktioniert identisch zu „doc_title_from_otc.js“, mit dem Ziel, das Feld „DCC description“ mit dem Inhalt der DCC-Beschreibung zu setzen. Ist der DCC leer, wird das Feld nicht gesetzt.

### non_physical.js

Anhand des OTCs wird in diesem Skript das Feld „Object Type“ gesetzt.
Der OTC wird mit der JavaScript Methode „search“ [17] und einem regulären Ausdruck als Argument durchsucht.
Fünf Zahlen als OTC beschreiben ein „non-physical“ Objekt, sobald er mit einem Buchstaben beginnt, ist der Objekt Typ „physical“.

### issued.js

Durch dieses Skript werden die Felder aus dem „issued for“-Block gesetzt. Nach einer, mit dem Betreuer Paul-Heinz Esters erarbeiteten Logik, wird der „Issue“ des jeweiligen Dokuments gesetzt.
Zu Beginn werden alle Felder auf die Option „no“ gesetzt.
„as built“ erhalten alle Dokumente mit DCC „QC05*“, wobei das Sternchen als Platzhalter dient. „DC08*“, „DC03*“ und „DC18*“ sind „issued for approval“, „QC04*“ und „QC1**“ „issued for information“.

## 5.3. Multiplizierungsalgorithmus

Der Sinn dieser Routine ist es, dem Benutzer bzw. Verwalter des MTP in Polarion die Arbeit abzunehmen, grundsätzlich immer gleiche DCCs für physische Objekte der Anlage einzupflegen. Der Nutzer soll schlussendlich einen fast fertigen MTP vor sich haben, bei dem es nur noch einigen Änderungen bedarf.
Grundsätzlich wird in dieser Implementierung auf das Überprüfen von mehrfach vorhandenen oder bereits verarbeitete Workitems verzichtet. Die Vervielfältigung soll zu Beginn eines Projekts einmalig stattfinden. Änderungen bzw. Ergänzungen werden von Hand gemacht.
Zudem soll es bei identischen Objekten auf der Anlage das zugehörige Prüfprotokoll nur einmal übergeordnet für alle Objekte geben, das Abnahmezertifikat aber für jedes einzelne Objekt. Das würde bedeuten, die vervielfältigten Workitems durch eine Routine der Anforderung nach zu reduzieren. Diese Routine wird hier nicht implementiert. Der Arbeitsschritt soll händisch erfolgen.

### 5.3.1. Wiki Page mit Velocity

Ein erster Ansatz wird auf Basis einer Wiki-Page mit der VTL erstellt. Die begrenzten Möglichkeiten der bereits veralteten Velocity Version sorgen jedoch für Probleme, weswegen der Ansatz verworfen wird. Bei dem Versuch mit Velocity wurde außerdem auf eine Informationsübrgabe von außen verzichtet, Parameter wurden „hardgecoded“. Die Grundlage des ersten Ansatzes, der Algorithmus, kann ohne viel Portierungsarbeit für den zweiten Ansatz weiterverwendet werden.

### 5.3.2. Plugin auf Basis des Beispiels „Servlet“

Zum Programmieren des Plugins wird die IDE[1] „Eclipse for Java Enterprise Developers“ verwendet. Die Einrichtung der Entwicklungsumgebung wird der Anleitung aus [13, Seite 4] entnommen. Statt wie in der Anleitung beschrieben muss im Feld „Runtime JRE“ nach vorangehender Installation „jre1.8.0_241“ statt „jdk-11.0.2“ gewählt werden.

#### Workitemtyp „properties“

Um dem User wie erwünscht die Möglichkeit zu geben, die Parameter des Vervielfältigungsalgorithmus zu verändern, wurde ein neuer Workitem-Typ „properties“ erstellt. Dieser beinhaltet folgende Custom-Fields:

- otcs for c&p objects, type=string
- dccs list for non c&p objects, type=string
- dcc list for c&p objects, type=string
- project ids, type=string
- dcc count for offsite cp, type=integer
- dcc count for offsite noncp, type=integer
- optional items list c&p, type=string
- optional items list non c&p, type=string

Über diese Felder legt der Benutzer fest, welche OTCs ein Leittechnikobjekt (c&p object) beschreiben, welche DCCs ein Leittechnikobjekt bzw. nicht-Leittechnikobjekt erhält, welche Projekte von der Vervielfältigung betroffen sein sollen, wie viele DCCs eines Leittechnikobjektes bzw.
nicht-Leittechnikobjektes off-site bzw. on-site sind und welche DCCs nur optional sind.
In den beiden Integer-Feldern, wird die Anzahl der off-site-DCCs gespeichert. Diese werden in der späteren Routine zuerst abgearbeitet. Ist die, in den Feldern gespeicherte Anzahl, der DCCs abgearbeitet, sind alle folgenden Dokumente für on-site. Die Grundlage des Testprojektes, sind die Listen aus Anhang A.1. Somit lassen sich alle Felder außer „otcs for c&p objects“ und „project id“

[1] integretated development environment

befüllen. „project id“ entält in diesem Fall nur den ID des Testprojekts. Aktuell ist nur ein Projekt in diesem Feld möglich. Die Liste der OTCs, die ein Leittechnikobjekt beschreiben wurde aus [18] entnommen.

### Beispielplugin „Servlet“

Die Beispielerweiterung von Polarion selbst wird von [13, Seite 8] wie folgt beschrieben: „This example allows you to create an extension for Wiki pages in form of creating a custom servlet to inform users, e.g. about statistics at the Home or Dashboard. The result will be your own servlet with a title and body represented by ‚.jsp‘ page (written by you) emebedded into a Wiki page.“
Eine ‚.jsp‘-Datei, ‚.jsp‘ steht für „Java Server Pages“, ist eine HTML[2]-Datei, die zusätzlich „Java“-Code enthält und somit dynamische Webseiten ermöglicht. [19][vergleiche] Die Klasse des Beispiels erweitert die Javaklasse „HttpServlet“ und überschreibt die Methoden „doPost()“ und „doGet()“. Im Funktionsrumpf von „doGet()“ wird der Beispielcode ausgeführt, in „doPost()“ wird „doGet()“ aufgerufen. Am Ende der „doGet()“ Methode werden Informationen an die ‚.jsp‘ Datei übergeben. Durch den Aufruf dieser Datei mit

```
<iframe width= 100% height= 200 src=/polarion/example/
frameborder=0 ></iframe>
```

im Code der Wiki-Page, wird der Inhalt der Datei dargestellt. In der ‚.jsp‘-Datei können sowohl HTML als auch Polarion-Makros verwendet werden. Diese Informationen können dem Quellcode der Erweiterung entnommen werden. Dieser ist nicht öffentlich zugänglich, jedoch im Verzeichnis jeder Polarion-installation vorhanden.
Auf Basis dieser Erweiterung wird ein eigenes Plugin entwickelt, die Grundfunktionen bleiben erhalten.

[2]Hypertext Markup Language

### Plugin „WorkItemsMultiply“

In folgendem Klassendiagramm sind die implementierten bzw. abgeänderten Klassen zu sehen. „ServiceServlet“ implementiert die selbe Grundfunktion wie die Klasse „CurrentUserWorkloadServlet“ des Beispiels, jedoch wurde der Code im „doGet()“ Funktionsrumpf verändert.

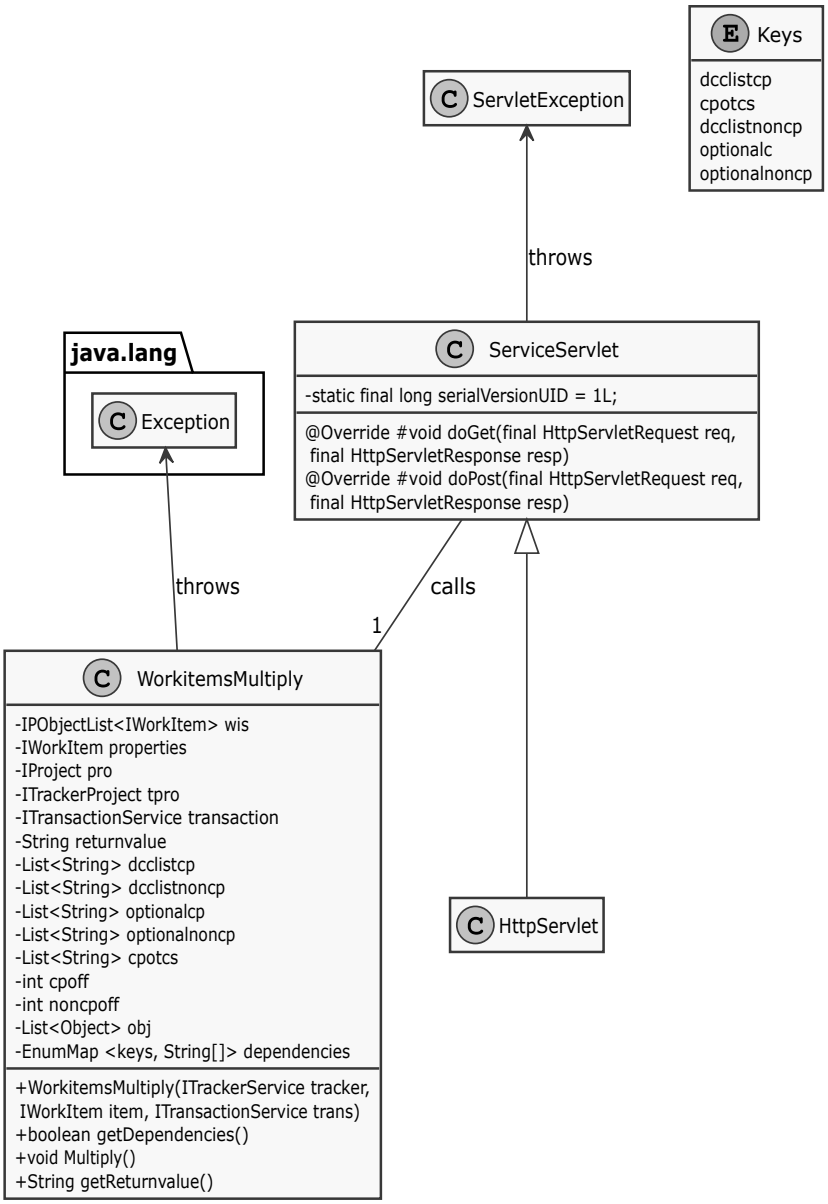


Abbildung 5.2.: Klassendiagramm der „WorkItemsMultiply“ Erweiterung

Die Funktionweise der Erweiterung wird anhand der einzelnen Klassen kurz erläutert. Alle Funktionen der API-Klassen können im Verzeichnis unter [20] gefunden werden. Die Stringvariablen „returnvalue“ und „returnvalue[0]“ der beiden Klassen werden kontinuierlich mit Debug-Informationen bzw. anwenderrelevanten Daten gefüllt und durch HTML formatiert. Diese Daten werden durch einfaches Ausgeben eines kombinierten Strings in „rendered.jsp“ dem Benutzer dargestellt.

### ServiceServlet

Die Basisklasse der Erweiterung basiert, wie oben genannt, auf der Klasse des Polarion-Beispiels.
Sämtlicher Code wird, wie auch in der Beispielklasse, in der „doGet()“ Methode der Klasse ausgeführt.

**doGet** Um den in der ‚.jsp' Seite dargestellten Text in richtiger Programmablaufreihenfolge darzustellen, wird des Öfteren der „Getter" „getReturnvalue" des „WorkItemsMultiply"-Objekts aufgerufen, anders als dargestellt.
In folgender Grafik wird „Beenden" bewusst in Anführungszeichen dargestellt, da ein, durch „return", Beenden in diesem Fall nicht möglich ist. Die Funktion wird dadurch beendet, dass Daten an die ‚.jsp'-Datei übergeben werden. Statt also zu beenden, werden durch „if"-Anweisungen die Aufrufe der „WorkItemsMultiply"-Funktionen verhindert und dementsprechende Informationen an „rendered.jsp" übergeben. Auch der Aufruf der „Multiply"-Funktion kann fehlschlagen, allerdings wird danach sowieso die „doGet"-Funktion „beendet".

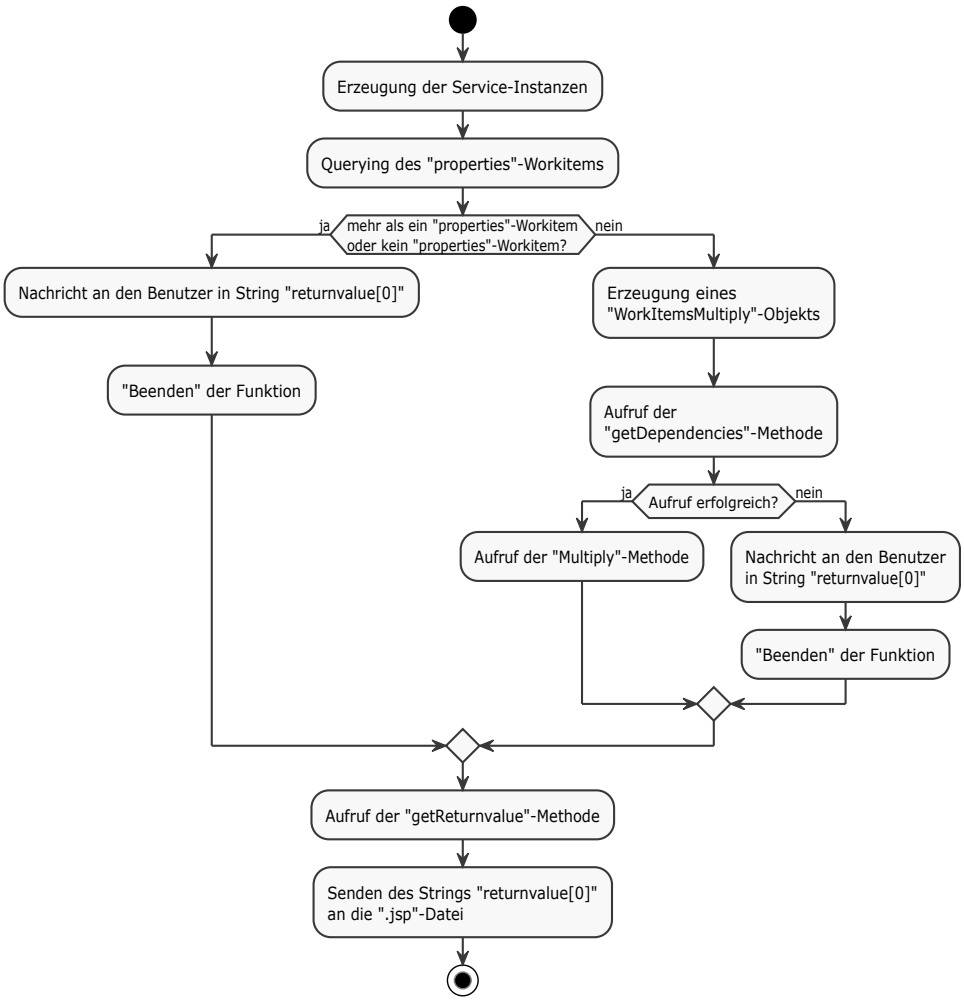


Abbildung 5.3.: Ablaufdiagram der „doGet"-Methode

### WorkItemsMultiply

Im Konstruktor der Klasse werden die grundlegenden Klassenvariablen initialisiert.

**getDependencies** Sollte ein Feld des „properties“-Workitems leer sein, gibt die Funktion ein „false“ zurück und „doGet()“ übergibt die Information, mit Verweis auf das erste leere Feld, an die ‚.jsp‘ Datei. Das Einlesen bzw. Speichern der Parameter aus den Feldern wurde durch ein „Enum“ und eine „EnumMap“ realisiert. Das sorgt für kompakteren und übersichtlicheren Code.

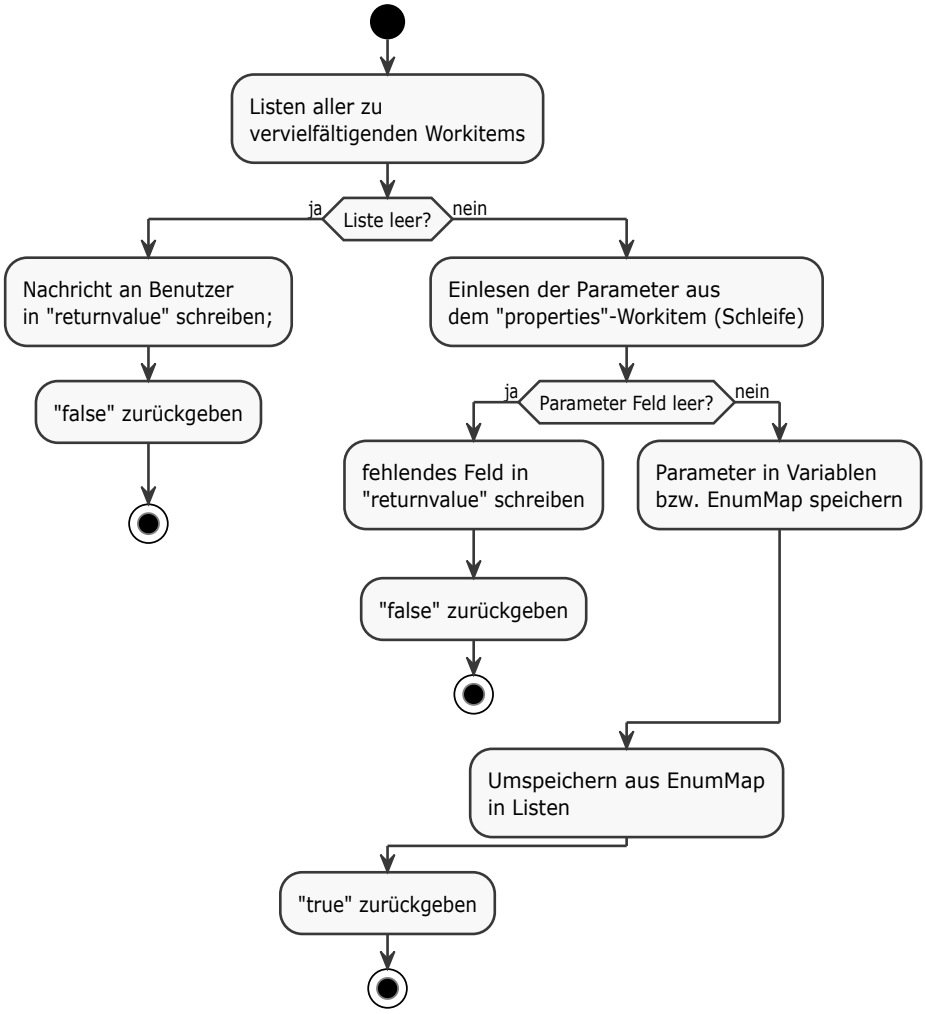


Abbildung 5.4.: Ablaufdiagramm der „getDependencies“-Methode

**Multiply** Die Vervielfältigungsalgorithmen für die zwei Arten von Objekten, unterscheiden sich nur durch die Anzahl der DCCs, der Art der DCCs und der Optionalität. Die in „getDependencies“ gespeicherten Parameterdaten werden je nach Objektart für die Routine genutzt.
Aktuell wird bei jedem Workitem das Feld „Code Letter“ auf die Option „E“ gesetzt.
Die Funktionen des „TransactionService“ können bei Fehlschlagen eine „java.lang.Exception“ werfen, die von der höheren Instanz „ServiveServlet“ gefangen und verarbeitet wird.

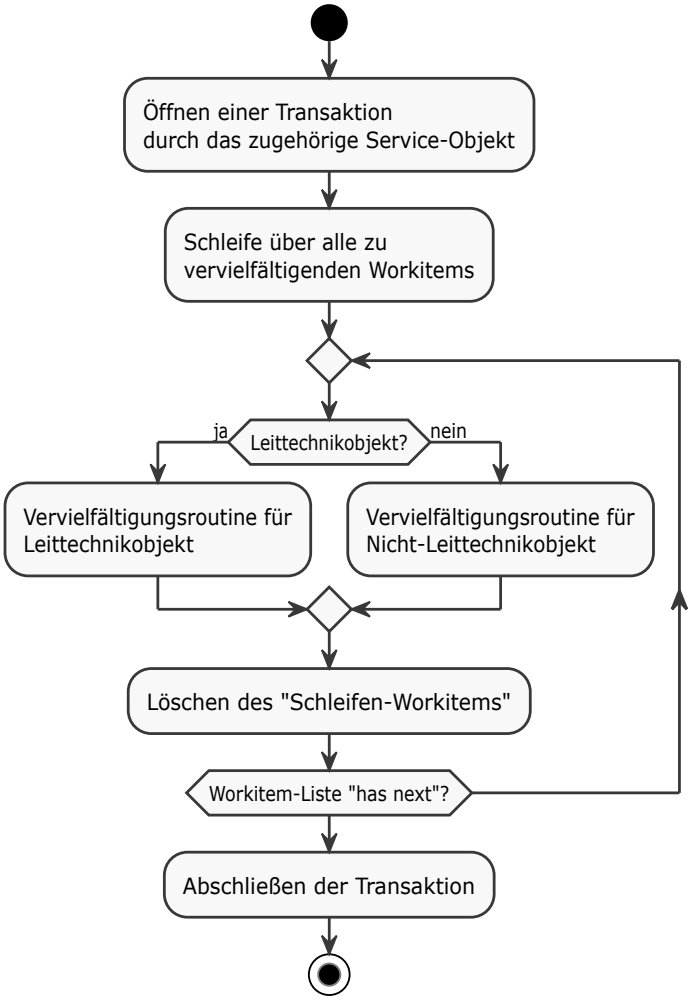


Abbildung 5.5.: Ablaufdiagramm der „Multiply“-Metode

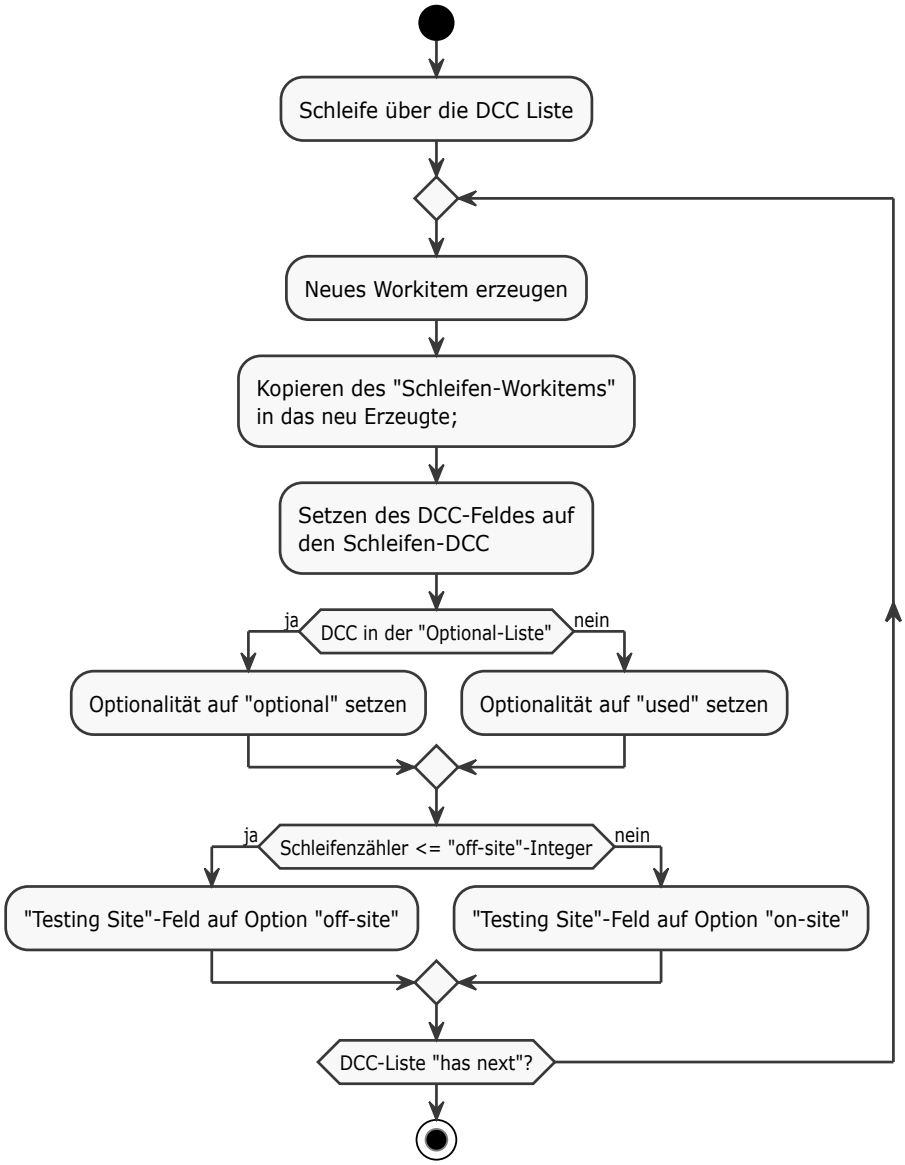


Abbildung 5.6.: Ablaufdaigramm der Vervielfältigungsroutine, fallunabhängig

### „rendered.jsp“

In der hier verwendeten Java Server Page, wird nur das, aus „ServiceServlet.doGet()“ übergebene, „Object“-Array angenommen, gecastet und als einfacher String ausgegeben. Die Formatierung dieses Strings findet in den Methoden der Klassen statt.

```
<%@ page import="java.util.Collection" %>
<%@ page import="java.util.Iterator" %>
<%@ page import="com.polarion.alm.tracker.model.IWorkItem" %>

<%
      Object[] info = (Object[])request.getAttribute("properties");
    String display = (String) info[0];
%>
Return: <%=display%>
```

Abbildung 5.7.: rendered.jsp

## 5.4. Importmasken

Die Objekte der beiden Aspektschlüssel „P“ und „L“ sollen aus PAM importiert werden. Dafür muss eine Import/Export-Maske festgelegt werden. Aufgrund der JavaSkripte aus Kapitel 4.4 und dem Plugin aus Kapitel 4.5 kann der Import auf ein Minimum reduziert werden. Folgende Informationen pro Objekt müssen aus PAM als Import zur Verfügung stehen.

- Projekt ID
- Seriennummer
- Dokument Typ
- Aspektschlüssel
- Referenzkennzeichen
- OTC mit optionalem Subtypen
- Hersteller
- zusätzliches optionales Ortskennzeichen

## 5.5. Bedienungsanleitung

Nach allen Tests und Rücksprachen mit dem Betreuer, wird eine kurze Bedienungsanleitung zur Verwendung des neuen Workitemtyps und des Plugin angefertigt. Diese beinhaltet den Arbeitsablauf, die zu erledigenden Arbeitsschritte, die benötigten Importdaten und kurze Erklärungen zu den Skripten und dem Plugin.

# 6. Tests

Das, auf die Implementierung folgende, Testen wird aufgrund von Covid-19 mit reiner „Dummy-Data“ durchgeführt.
Dafür wurden sechs, auf Basis der Excel-MTP Vorlage erstellte, Workitems durch die Importfunktion in das Polarion Projekt importiert. Um später das Plugin zu testen, werden jeweils zwei Workitems jedes relevanten Aspektschlüssels importiert. Während des Imports sind neben einigen Grundparametern, wie, ab welcher Excel-Zelle der Import beginnen soll, auch ein „Mapping“ festzulegen, das heisst vorallem bei Enum-Felder muss festgelegt werden, welche Option der Zelle in Excel der „Enum-Option“ des Feldes in Polarion entspricht. Das Mapping kann in eine Konfiguration gespeichert, um so später nocheinmal verwendet zu werden. In Anhang 4.3 ist ein Teil des Mappings und der Importvorschau zu sehen.
Um die Funktionalität der JavaScript-Skripte zu testen, werden die importierten Workitems abgeändert und gespeichtert. Zusätzlich werden einige Sonderfälle erzwungen, um bei der Implementierung vergessene Möglichkeiten zu überprüfen.
Nach dem Import und Testen der Daten wird durch Aufrufen einer Wiki-Page, mit der in Kapitel 5.3.2 genannten Codezeile als Inhalt, das Plugin aufgerufen. Nach einiger Rechenzeit wird auf der Seite der Rückgabetext angezeigt.
Im Prinzip wurde der Arbeitsablauf nach Abbildung 4.1 getestet, nur ohne reale Importe.

# 7. Ergebnisse

Während des Imports stellt sich heraus, dass die JavaScript-Skripte auch während der Erzeugung der Vorschau abgearbeitet werden. Das erspart dem Anwender einige Mühen, da nicht der gesamte Import fehlschlägt, sondern lediglich die Bildung der Vorschau. So kann schnell entweder das „Mapping“ oder die Importdatei geändert werden. Schlägt der gesamte Import fehl, muss die Rückgabeinformation der Skripte einer unübersichtlichen „log“-Datei entnommen werden. Zudem zeigt sich, dass Polarion verlangt, dass entweder das Titel- oder Beschreibungsfeld bei einem Import angegeben werden muss. Beide werden aber durch Skripte automatisch befüllt. So muss „Dummy“-Information in eines der beiden Felder „gemappt“ werden.
Die durch das Plugin neu erstellten Workitems können in der Polarionbenutzeroberfläche (Abbildung A.5) betrachetet werden. Workitems des Aspektschlüssels „C“ bleiben wie erwartet vom Plugin unberührt.

**Probleme** Anders als in der Anleitung [13, Seite 5] kann das eigen entwickelte Plugin nicht in die installierte Polarion Anwendung importiert werden. Das Plugin funktioniert nur dann, wenn Polarion durch die IDE, wie in [13, Seite 6] beschrieben, gestartet wird. Jedoch funktioniert auf diesem Weg die „FMC“-Erweiterung nicht. Somit konnte nicht getestet werden, ob, wie erwünscht, das Speichern eines Workitems im Code des Plugins auch eine Abarbeitung der JavaSkripte zur Folge hat.
Zugleich scheint das Testen des Plugins „Datenleichen“ zu erzeugen, die zwar in der Polarionoberfläche augenscheinlich vorhanden sind und durch API-Funktionen „gequeried“ werden können, aber bei Zugriff über beide Möglichkeiten einen Fehler erzeugen und die Exception `com.polarion.platform.persistence.UnresolvableObjectException` werfen. Im Revisionsverzeichnis aller Datensätze in Polarion werden diese Workitems allerdings nicht angezeigt. Polarion verfügt über eine „Maintenance“-Funktion um einige bekannte Fehler zu beheben, Chache zu leeren oder „log“-Datein zu erzeugen. Diese Funktion behebt das Problem nicht.

Möglicherweise werden diese erzeugt, wenn Objekte, die im Plugin noch referenziert sind, über die Oberfläche gelöscht werden. Das ist während des Testens des Öfteren passiert. Durch „null“-Setzen der Objektreferenzen im Plugin und dem darauffolgenden Anstoß des Java „Garbage-Collector“ wird versucht dieses Problem zu lösen. Da bis zu diesem Zeitpunkt schon unübersichtlich viele „untote Workitems“ enstanden sind, ist es unmöglich zu sehen, ob dies das Problem löst. Diese „toten Workitems“ konnten nicht wieder aus Polarion entfernt werden. Vielleicht liegt das Problem auch an völlig anderer Stelle. Kollegen kennen diesen Fehler nicht und der Polarion-Support ist, trotz Siemens-Zugehörigkeit, kaum zu Erreichen. Die Fehlermeldung ist in Anhang A.6 abgebildet.

# 8. Zusammenfassung, Fazit und Ausblick

## 8.1. Zusammenfassung

Das Ziel der Arbeit war es, den MTP sinnvoll in das bestehende Online-Tool Polarion zu übertragen. Nach kurzer Einarbeitung in das Themengebiet und einigen Fachgesprächen wurde der erste Entwurf ausgearbeitet. Zu Beginn wurden alle benötigten Felder und Variablen des neuen Workitemtyps definiert. Während der Erstellung dieser Eingabemasken kam die Idee auf, JavaScript-Skripte einer Erweiterung zu verwenden, die dem Benutzer die Bedienung der Oberfläche und vorallem den Datenimport deutlich vereinfachen. Da durch die globale Covid-19 Situation die innerbetrieblichen Abläufe stark ausgebremst sind, können einige Datenstandards nur bedingt festgelegt werden. Nach zügigem Voranschreiten der Implementierung ergab sich in Gesprächen die Idee den MTP eines jeden neuen Projekts fast automatisch zu erzeugen. Durch einen Vervielfältigungsalgorithmus sollen zu jedem physikalischen Objekt, das später auf der Anlage verbaut wird, automatisch die benötigten Dokumente gebildet werden. Ein erster Ansatz auf Basis einer Wiki-Page mit VTL wurde nach Bemerken offensichtlich fehlender Funktionen verworfen. Der zweite Ansatz basierte auf einer Beispiel-Erweiterung, die im Verzeichnis einer jeden Polarion Installation gefunden werden kann. Dieses Plugin erweitert durch eine Java-Klasse die Klasse „Http-Servlet“ und wird durch eine Wiki-Page aufgerufen. Durch Abändern eines Methodenrumpfes wurde eine eigene Funktionalität geschaffen, der Aufruf des Plugins bleibt jedoch unverändert. Abschließend wurden Importmasken definiert, die die Mindestanforderung an Daten pro Objekt festlegen. Da aufgrund der oben genannten Virus-Pandemie keine realen Daten zur Verfügung standen, wurden einige „Dummy“-Objekte angelegt, jedoch ausreichend genug, um den vollen Funtionsumfang zu Testen. Nach erfolgreichem Testen wurde eine kurze Funktionsanleitung verfasst.

## 8.2. Fazit

Mir persönlich machte vorallem die Arbeit aus dem Homeofice zu schaffen, da trotz richtigem Heim-Arbeitsplatz ständig Ablenkungen präsent waren. Grundsätzlich wurden dennoch alle Punkte des Arbeitsauftrages implementiert und funktionieren fast vollständig. Die auftretenden Probleme konnte ich leider nicht lösen.
Ich fand auch die Polarion Handbücher nicht ausreichend detailliert beschrieben. Zur Erstellung völlig eigener Erweiterungen fehlen meiner Meinung nach auch die Erläuterungen zu Einstiegspunkten der API bzw. der von Polarion aufgerufenen Systemfunktion. Auch die Fehler in den Handbüchern bzw. das nicht-Funktionieren dort eindeutig als durchführbar beschriebener Arbeitsabläufe ärgert mich.
Polarion ist, in meinen Augen, ungeeignet große Datenmengen in meinem eigens konzipierten Plugin zu verarbeiten. Die erstellten Test-Workitems, von denen nur vier verarbeitet wurden, hatten eine Prozessdauer von ca. 30 Sekunden. Eine Anlage von normaler Größe hat in etwa 500-1000 Komponenten. Die hochgerechnete Prozessdauer hierbei überschreitet meiner Meinung nach die Grenze des effektiven Arbeitens. Außerdem haben großen Importe nach Polarion auf meinem lokalen System ein „Überlaufen“ des Arbeitsspeichers zur Folge. Inwieweit das Problem durch Zuweisung weiteren Arbeitsspeichers gelöst werden kann, oder ob das Problem auch bei Abarbeitung meines Plugins auftritt, konnte ich nicht feststellen.

## 8.3. Ausblick

Eine mögliche, anschließende Thematik zu dieser Arbeit, bedingt durch die aufgetretenen Leistungsprobleme wäre, die Funktion des Plugins und der JavaScript Skripte in ein eigenes Tool auszulagern.

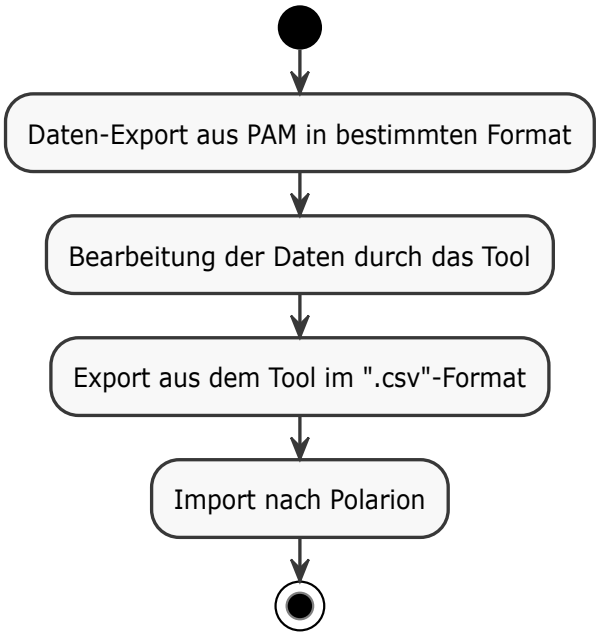


Abbildung 8.1.: Möglicher, weiterführender Workflow

Dieses könnte mit einer grafischen Oberfläche und weiteren Funktionen, wie z.B. der Überprüfung auf bereits abgearbeitete Objekte, erarbeitet werden. Auch die in Kapitel 5.3 kurz beschriebene, aber aufgrund von fehlender Leistung nicht implementierte, Routine zur Datensatzaufbereitung durch Reduzierung der Workitems auf ein Minimum wäre umsetzbar.
Allgemein wäre eine Überarbeitung der JavaScript Skripte möglich, falls sich bei der Verwendung dieser Arbeit Anwendungsfehler finden oder Anforderungen ändern.
Ferner bleibt abzuwarten, wie sich die Umgewöhnung der Kollegen auf das neue Tool gestaltet.

# Literaturverzeichnis


[1] "Polarion," 3/18/2020. [Online]. Available: https://www.competence-site.de/polarion-software-gmbh/ 1, 2.4

[2] R. Paschotta. [Online]. Available: https://www.energie-lexikon.info/hochspannungs_gleichstromuebertragung.html 2.1

[3] P.-H. Esters, "Mögliche Themen für eine Bachelorarbeit (intern)." 2.2

[4] ——, "MTP-Konzept (intern)." 2.1

[5] Interessengemeinschaft Energieverteilung, "Elektrische Energieübertragung und -verteilung - Stationen Kennzeichnung und Dokumentation: Teil 1: Strukturierungsprinzipien und Referenzkennzeichnung nach IEC 81346." [Online]. Available: https://www.igevu.de/files/1_refkennz_evu_2011_05_18.pdf 2.3

[6] ——, "Elektrische Energieübertragung und -verteilung - Stationen Kennzeichnung und Dokumentation: Teil 2: Kennzeichnung und Ordnung der Dokumentation nach IEC 61355." [Online]. Available: https://www.igevu.de/files/2_dokstruct_evu_2010_08_01.pdf 2.3, 2.6

[7] E. Moehrlein, "Dokumentationssystem IEC / OTC (intern)." 2.3, 2.3.1, 2.3, 2.4, 2.5, 2.7, 2.8

[8] W. Kurze, "TC_29102_H002_BBT010_02 Design Guide Testing HVDC 1.0 Training (intern)." 2.2

[9] P.-H. Esters, "TC_29102_H002_BBT010_02 Design Guide Testing HVDC 2.0 Training (intern)." 2.3.2

[10] Polarion, "User and Administration Help: Advanced querying." [Online]. Available: https://almdemo.polarion.com/polarion/help/index.jsp?topic={%}2Fcom.polarion.xray.doc.user{%}2Fguide{%}2Fxid1570724.html 2.4.1

[11] "Apache Velocity Engine - User Guide," 2020-04-04T01:31:08.000Z. [Online]. Available: https://velocity.apache.org/engine/1.7/user-guide.html#what-is-velocity 2.4.2

[12] Polarion, "Advanced Wiki Training _ Training Materials _ Documents & Pages _ PolarionTraining (intern)." 2.4.2

[13] ——, "Polarion SDK Dokumentation." [Online]. Available: https://almdemo.polarion.com/polarion/sdk/doc/sdk.pdf 2.4.3, 2.4.3, 2.4.3, 2.4.3, 2.4.3, 5.3.2, 5.3.2, 7

[14] "Polarion Extensions," 2020-06-29T13:26:35.000Z. [Online]. Available: https://extensions.polarion.com/extensions/134-fmc-work-item-save 2.4.4, 5.2

[15] Polarion, "Polarion API." [Online]. Available: https://almdemo.polarion.com/polarion/sdk/doc/javadoc/com/polarion/platform/persistence/model/IPObject.html#setEnumerationValue(java.lang.String,java.lang.String) 5.2

[16] w3schools, "https://www.w3schools.com/jsref/jsref_replace.asp." [Online]. Available: https://www.w3schools.com/jsref/jsref_replace.asp 5.2

[17] ——, "JavaScript String search() Method." [Online]. Available: https://www.w3schools.com/jsref/jsref_search.asp 5.2

[18] Moosburger, "Object Type Catalouge(OTC) (intern)." 5.3.2

[19] DateiEndung.com, ".jsp Dateiendung." [Online]. Available: https://www.dateiendung.com/format/jsp 5.3.2

[20] Polarion, "Polarion API: All Classes." [Online]. Available: https://almdemo.polarion.com/polarion/sdk/doc/javadoc/allclasses-frame.html 5.3.2

# A. Anhang

## A.1. Dokumentationsschema

SIEMENS

Project Documentation and Reference Designation System
Testing Documentation for C&P Equipment (DCC)

| Program | Protocol | Remark | DocTree |
|---|---|---|---|
| **Offsite Workshop / Factory Acceptance Tests** | | | |
| DC080 Inspection and test plan (ITP) | QC010 Declaration of Conformity | Acc IEC 17050<br>Required for CE conformation | |
| | QC020 Inspection certificate | Internal C&P acc. OTC | O2 |
| | QC040 Test certificate (workshop) | C&P acc. product key | O2 /E4 |
| DC081 Inspection and test plan (type test) | QC041 Test certificate (type test) | Test plan and certificate acc. OTC | E4 |
| DC082 Inspection and test plan (routine test) | QC042 Test certificate (routine test) | RTP / Routine Test (electrical)<br>Test plan acc. OTC<br>Test certificate acc. product key | E4 |
| DC083 Inspection and test plan (installation test) | QC043 Test certificate (installation test) | Installation Test (mechanical, visual)<br>Test plan acc. OTC<br>Test certificate acc. product key | E4 |
| DC085 Inspection and test plan (local test) | QC045 Test certificate (local test) | Optional C&P (local test lab)<br>Test plan acc. OTC<br>Test certificate acc. product key | E4 |
| DC086 ... DC089 | QC046 ... QC049 | spare | |

Abbildung A.1.: Schema zur Dokumentation für Leittechnik Objekte off-site

**Project Documentation and Reference Designation System**
**Testing Documentation for C&P Equipment (DCC)**

SIEMENS

| Program | | Protocol | | Remark | DocTree |
|---|---|---|---|---|---|
| **Onsite / Site Acceptance Tests** | | | | | |
| DC030 | Instruction for start up | QC050 | Acceptance report | Optional combined pre-commissionig and sub-system Test<br>Instruction acc. OTC<br>Report acc. product key | F2 |
| DC031 | Instruction for start up (Pre-commissioning) | QC051 | Acceptance report (Pre-commissioning) | Instruction acc. OTC<br>Report acc. product key | F1 |
| DC032 | Instruction for start up (Sub-system) | QC052 | Acceptance report (Sub-system) | Instruction acc. OTC<br>Report acc. product key | F2 |
| | | QC056 | Site Acceptance Civil | acc. location key<br>Site Acceptance Certificate (Civil) | SA1 |
| | | QC057 | Site Acceptance Installation | acc. location key<br>Site Acceptance Certificate (Installation) | SA2 |
| DC038 | Instruction for start up (Installation Test) | QC058 | Acceptance report (Installation) | Instruction acc. OTC<br>Report acc. product key | F1 |
| DC036, DC037, DC039 | | QC059 | | spare | |

Abbildung A.2.: Schema zur Dokumentation für Leittechnik Objekte on-site

# Project Documentation and Reference Designation System
# Testing Documentation for Equipment (none C&P) (DCC)

SIEMENS

| Program | | Protocol | | Remark | DocTree |
|---|---|---|---|---|---|
| **Offsite Workshop / Factory Acceptance Tests** | | | | | |
| DC080 | Inspection and test plan (ITP) | QC040 | Test certificate (type and routine test) | Standard combined for type test and routine test | E2 |
| DC081 | Inspection and test plan (type test) | QC041 | Test certificate (type test) | Optional separate type test doc. | E2 / E3 |
| DC082 | Inspection and test plan (routine test) | QC042 | Test certificate (routine test) | Optional separate routine test doc. | E2 / E3 |
| DC083 | Inspection and test plan (installation test) | QC043 | Test certificate (installation test) | Plan acc. OTC<br>Certificate acc. product key | E2 / E3 |
| DC085 | Inspection and test plan (local tests) | QC045 | Test certificate (components) | Optional HV Test of equipment at local test lab<br>Plan acc. OTC<br>Certificate acc. product key | E3 |
| DC086 ... DC089 | | QC046 ... QC049 | | spare | |

Abbildung A.3.: Schema zur Dokumentation für nicht-Leittechnik Objekte off-site

# Project Documentation and Reference Designation System
# Testing Documentation for Equipment (none C&P) (DCC)

SIEMENS

| Program | | Protocol | | Remark | DocTree |
|---|---|---|---|---|---|
| **Onsite / Site Acceptance Tests** | | | | | |
| DC030 | Instruction for start up | QC050 | Acceptance report | Optional combined pre-commissionig and sub-system Test | F2 |
| DC031 | Instruction for start up (Pre-commissioning) | QC051 | Acceptance report (Pre-commissioning) | Instruction acc. OTC<br>Report acc. product key | F1 |
| DC032 | Instruction for start up (Sub-system) | QC052 | Acceptance report (Sub-system) | Instruction acc. OTC<br>Report acc. product key | F2 |
| DC035 | Instruction for start up (HV Tests) | QC055 | Acceptance report (HV Tests) | Optional HV Test of equipment onsite<br>Instruction acc. OTC<br>Report acc. product key | F1 |
| | | QC056 | Site Acceptance Civil | acc. location key<br>Site Acceptance Certificate (Civil) | SA1 |
| | | QC057 | Site Acceptance Installation | acc. location key<br>Site Acceptance Certificate (Installation) | SA2 |
| DC038 | Instruction for start up (Installation Test) | QC058 | Acceptance report (Installation) | Instruction acc. OTC<br>Report acc. product key | F1 |
| DC036, DC037, DC039 | | QC059 | | spare | |
| DC180 | Instruction for inspection and test | QC150 | Acceptance Report (FQP) | Field Quality Plan (&C, &M, &E) | FQP (F1) |

Abbildung A.4.: Schema zur Dokumentation für nicht-Leittechnik Objekte on-site

## A.2. Polarion-Oberfläche

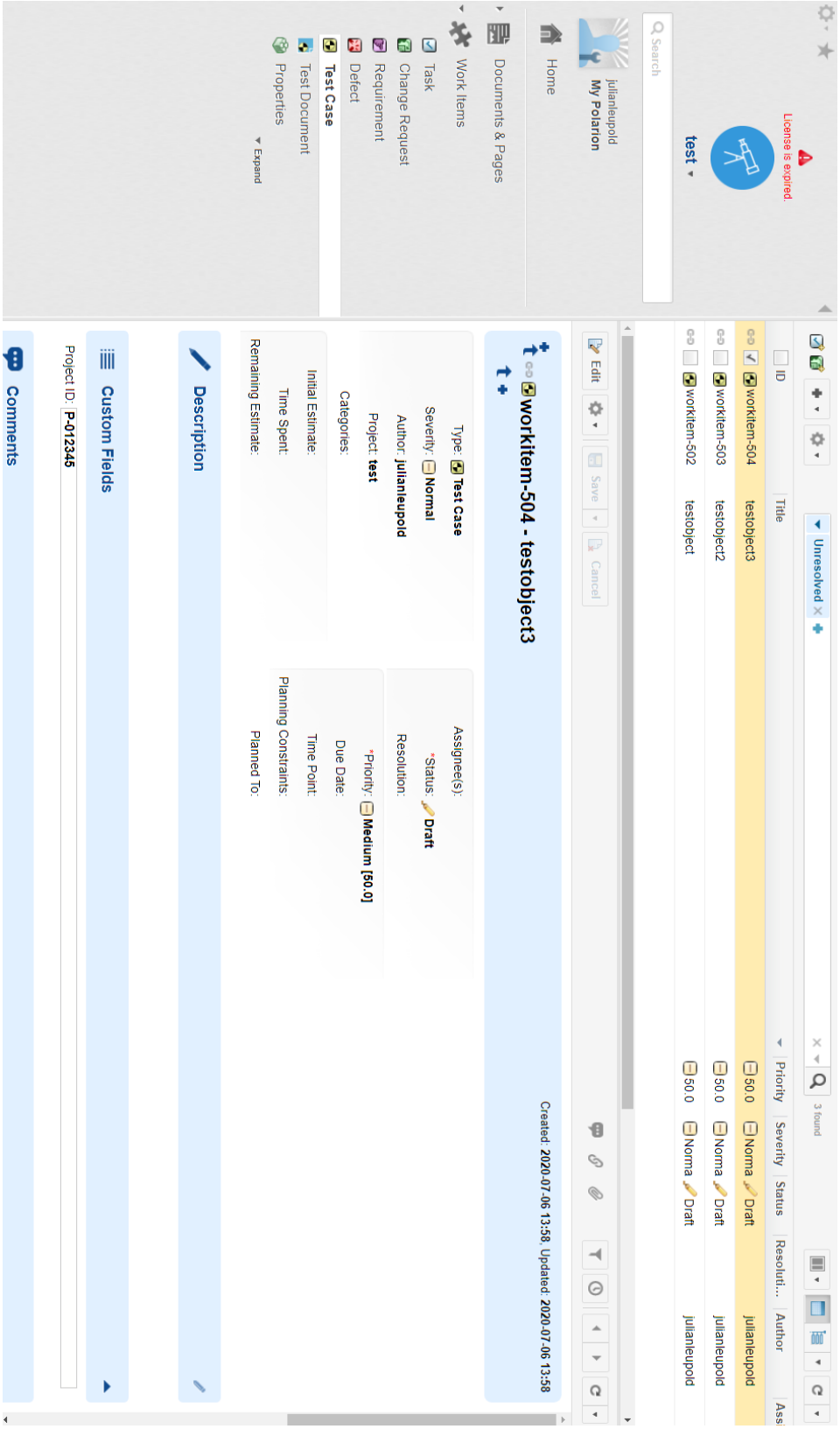


Abbildung A.5.: Gesamte Benutzeroberfläche

## A.3. Velocity und API Code-Beispiel

```
##
## start transaction
##

$transactionService.beginTx()

##
## create a new workitem
##
#set($newWi=$trackerService.createWorkItem($projectService.getProject($page
    .project)))

## set the workitem type
#set($enumTypes=$trackerService.getTrackerProject($page.project).
    workItemTypeEnum)
$newWi.setType($enumTypes.wrapOption("task"))

## set the title
$newWi.setTitle("Created by Velocity")

## set the description
#set($textObject=$objectFactory.newPlainText("This workitem was created by
    Velocity"))
$newWi.setDescription($textObject)

## save the workitem
$newWi.save()

##
## end transaction, persist the work item changes
##
$transactionService.commitTx()
```

Abbildung A.6.: Codebeispiel Velocity

Mit `#set()` werden Variablen gesetzt. Um eine Änderung an Workitems persistent zu machen, muss immer eine `transaction` begonnen und nach dem Speichern des Workitems beendet werden.
Die Extension um die Instanz `$objectFactory` zu verwenden, ist unter der verwendeten Polarionversion nicht mehr verfügbar.

## A.4. Interface Testdocument

**workitem-405 - Single Line Diagram, variant for FAT (SubSIM), P-012345_EC_30011.A002&EFA010_EN_00**

| | | |
|---|---|---|
| Type: **Test Document** | Assignee(s): | VSC: **xD** |
| *Severity: **Normal** | Resolution: | LCC: **xD** |
| Project: **test** | Custom ID: | Object Type: **Non Physical Object** |
| Project ID: **P-012345** | Internal remark: | DMS Identifier: |
| | | TestingSite: |

**Description**

Single Line Diagram, variant for FAT (SubSIM)

Documenttitle: **Single Line Diagram, variant for FAT (SubSIM)**
DCC description: **Overview Diagram**
Serialnumber:

**----------doc_numbering----------**

| | | |
|---|---|---|
| Document Type: **E** | Station number: | Level E: |
| Aspect Key: **C** | Level B: | Level F: |
| OTC.Subtype: **30011.A002** | Level C: | Level G: |
| | Level D: | Level H: |

| | |
|---|---|
| Code Letter: **&E** | DCC Class: **FA010** |
| OTC: **30011** | |
| Subtype: **.A002** | |

| | |
|---|---|
| OTC Code explicit for Location Aspect: | Level D, L: |
| Station number, L: | Level E, L: |
| Level B, L: | Level F, L: |
| Level C, L: | Level G, L: |
| | Level H, L: |

Abbildung A.7.: Testdocument Teil 1

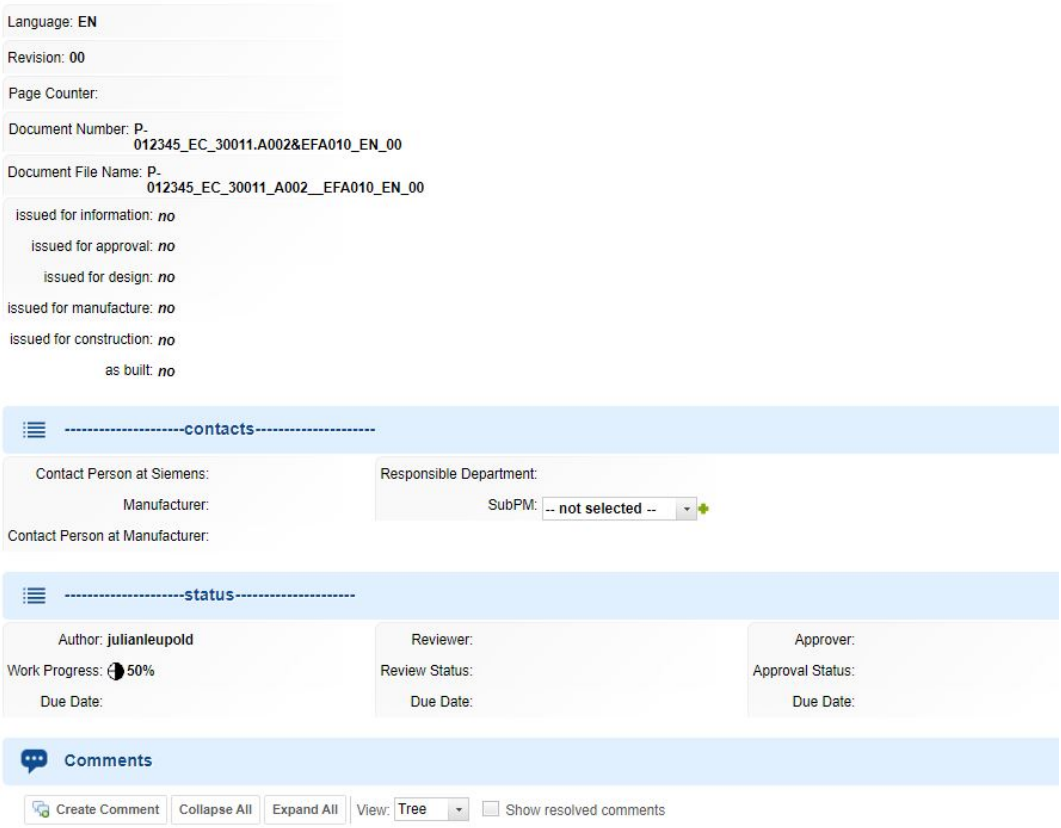


Abbildung A.8.: Testdocument Teil 2

## A.5. Importpreview

Abbildung A.9.: Preview eines Imports

## A.6. Fehlermeldung bei „untoten“ Workitems

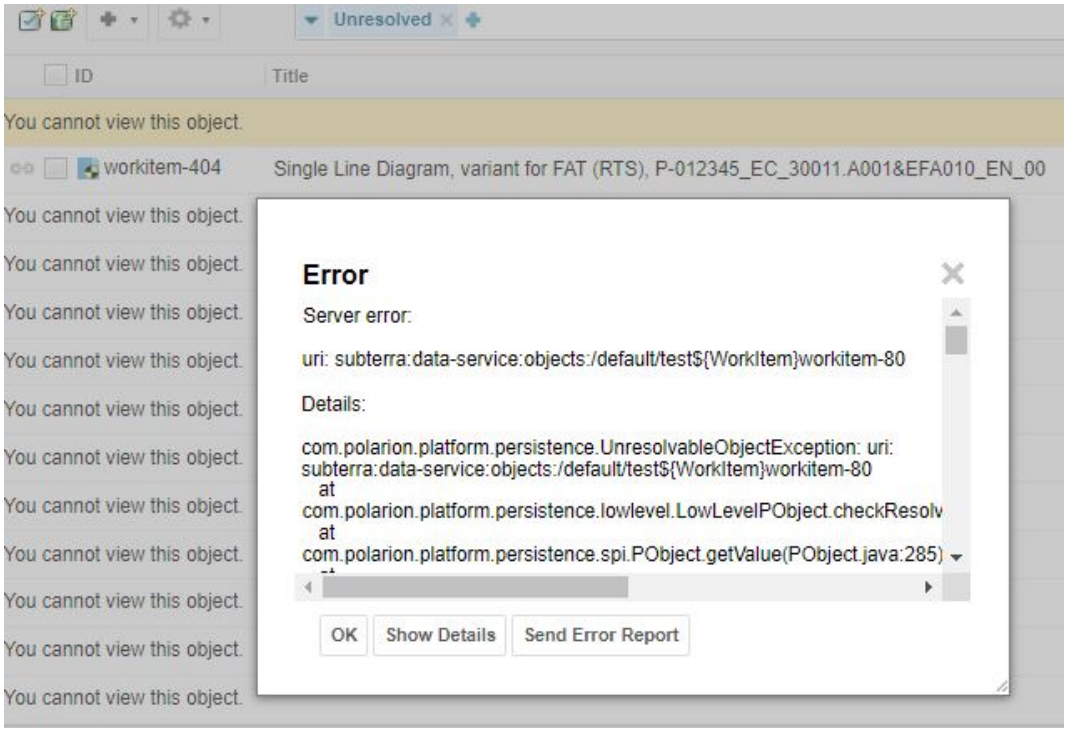


Abbildung A.10.: Fehlermeldung/Exceptionstack