



Dokumentation zur betrieblichen Projektarbeit

Autonummerierung

Fachinformatiker für Anwendungsentwicklung

Prüflingsname:	Allan Barry
Prüflingsnummer:	99008
Azubi-IdentNR:	1550003115913
Abgabedatum:	30.11.2021
Ansprechpartner:	Sylvie Loos

1. Inhaltsverzeichnis

Contents

1.	Inhaltsverzeichnis	2
2.	Abbildungsverzeichnis	4
3.	Einleitung	4
3.1	Vorstellung der Arsandis GmbH.....	4
3.2	Projektthema	4
3.3	Bearbeitungszeitraum	4
3.4	Rahmenbedingungen	4
3.5	Projektumfeld	5
3.5.1	Arsandis GmbH.....	5
3.5.2	Sensus	5
3.5.3	Pure Technologies Ltd	5
3.5.4	Systeme.....	5
3.6	Ziel & Nutzen des Projektes.....	6
3.7	Voraussetzungen für das Verständnis.....	6
3.7.1	Identifikationsattribute.....	6
3.7.2	Das WTPart-Konzept.....	6
3.7.3	Familientabellen.....	8
4.	Projektplanung	8
4.1	Projektphasen.....	8
4.1.1	Soll-Projektphasen.....	9
4.1.2	Ist-Projektphasen	9
4.1.3	Projektphasen-Vergleichs-Fazit.....	9
4.2	Projektkosten.....	10
4.2.1	Amortisation	11
5.	Definition der Anforderungen.....	11
5.1	Lastenheft.....	11
5.2	Ausgangszustand.....	11
5.3	Ist-Analyse	12
5.3.1	Hochgeladene CAD-Dateien.....	12
5.3.2	Assoziierte CAD-Dateien	12
5.3.3	Assoziation von WTParts während des Hochladens von CAD-Dateien	12
5.3.4	Assoziation von WTParts zu CAD-Dateien in Windchill	13
5.3.5	Händisch erstellte WTParts.....	13

5.4	Pflichtenheft	13
5.5	Soll-Analyse.....	14
5.5.1	Hochgeladene CAD-Dateien.....	14
5.5.2	Assoziierte CAD-Dateien	14
5.5.3	Assoziation von WTParts während des Hochladens von CAD-Dateien	14
5.5.4	Assoziation von WTParts zu CAD-Dateien in Windchill	14
5.5.5	Händisch erstellte WTParts.....	14
5.6	Wirtschaftlichkeitsanalyse.....	10
6.	Entwurfsphase.....	15
6.1	Zielplattform	15
6.1.1	Windchill PDMLink	15
6.1.2	LDAP	15
6.1.3	Datenbank.....	15
6.1.4	Webserver.....	15
6.1.5	Java (JRE).....	15
6.2	Entwurf für benötigte Systemkonfigurationen.....	16
6.2.1	Datenbank-Sequenz	16
6.2.2	Preferences	16
6.2.3	OIR (Object Initialization Rule)	16
6.2.4	Attribute	16
6.3	Entwurf für benötigte Java-Programm Logik.....	16
6.3.1	Benötigte Klassen	16
6.3.2	Grundlogik in PAP-Form	17
7.	Implementierungsphase	19
7.1	Erstellung einer Datenbank-Sequenz.....	19
7.2	Anpassung der Preferences	19
7.3	Anpassung der OIR.....	20
7.4	Erstellung der Attribute.....	22
7.5	Erstellung der Java-Klassen.....	23
8.	Qualitätssicherung.....	23
8.1	Testvorgang	24
8.2	Finaler Logiktest.....	24
8.3	Testfazit	25
9.	Projektabschluss.....	25
10.	Glossar	25
11.	Quellenverzeichnis.....	27

12. Anhang.....	27
12.1 DB-Sequenz-Template.....	27
12.2 OIR WTPart	27
12.3 ASEPMDocumentNamingDelegate	28
12.4 ASPartNumberingRule.....	30
12.5 Nummerierungstestplan	32

2. Abbildungsverzeichnis

Abb. 1: CAD-Struktur

Abb. 2: WTPart-Struktur

Abb. 3: WTPart-Struktur mit allen Abhängigkeiten

Abb. 4: PAP ASEPMDocumentNamingDelegate

Abb. 5: PAP Number Generator

Abb. 6: Preference Management

Abb. 7: Object Initialization Rule Administration

Abb. 8: Type and Attribute Management – AS CAD Document

Abb. 9: Type and Attribute Management – AS Part

3. Einleitung

3.1 Vorstellung der Arsandis GmbH

Bei der Arsandis GmbH handelt es sich um IT- und Prozessberatung in Angkofen, Pfaffenhofen an der Ilm. Schwerpunkt ist hier die Beratung von Fertigungsunternehmen im Bereich Product Lifecycle Management (PLM).

Weitere zukunftsorientierte Technologien werden ebenfalls verfolgt, darunter zum Beispiel AR-, VR-, und IoT-basierte Lösungen für die Industrie von morgen.

Als offizieller PTC-Partner liegt der Fokus hier meist auf dem PTC-Portfolio, welches ein umfangreiches Ökosystem für die Fertigungsindustrie bietet.

3.2 Projektthema

Das Thema der Projektarbeit ist die Autonummerierung von CAD-Dateien und WTParts (Produktrepräsentationen) nach einem festen Schema innerhalb eines PLM-Systems im Rahmen eines Kundenauftrags.

3.3 Bearbeitungszeitraum

Beginn: 04.10.2021

Ende: 18.10.2021

3.4 Rahmenbedingungen

Diese Projektarbeit wird als Teil eines Kundenprojekts umgesetzt. In dem Projekt geht es um Erweiterungen und Anpassungen eines bestehenden PLM-Systems (PTC Windchill PDMLink, web-basierte Browser-Applikation) umgesetzt. Dieses System wird seit einigen Jahren von der Firma Sensus genutzt. Das aktuelle Projekt wurde jedoch von der Firma Pure Technologies Ltd in Auftrag gegeben, welche das System künftig ebenfalls nutzen möchte. Dafür wird eine separate Organisationseinheit innerhalb des Systems angelegt, damit die beiden Unternehmen von einander getrennt arbeiten können. Bei Sensus und Pure Technologies Ltd handelt es sich um Töchter des

selben Unternehmens, weswegen wir die Projektanfrage für die Integration in ein bestehendes System bekamen. Zum heutigen Zeitpunkt befinden wir uns bereits in Phase 2 des Projektes.

In Phase 1 des Projektes wurden größtenteils Anpassungen des Change Management (Änderungsverwaltung) und des Promotion Process (Freigabeprozess) durchgeführt. Weitere Schwerpunkte der ersten Phase waren die Integration aller Benutzer, das Erstellen neuer Dokumenttypen, die Erweiterung und Anpassung der im System verfügbaren Attribute, das Erstellen der Arbeitsbereiche (Produkte und Bibliotheken) und die Definition sowie Umsetzung der Rechteverteilung.

In Phase 2 des Projektes behandeln wir das BOM-Management (Stücklistenverwaltung) und das CAD Data Management (CAD-Datenverwaltung). Sonstige Themen der Phase 2 sind weitere kleine Erweiterungen und Anpassungen des Change Management und des Promotion Process. Die Umsetzung meiner Projektarbeit findet im Rahmen des CAD Data Management der Phase 2 des Projektes statt.

3.5 Projektumfeld

3.5.1 Arsandis GmbH

Dieses Projekt führe ich von meinem Arbeitsplatz in den Büroräumen bei der Arsandis GmbH aus durch. Es handelt sich um ein Festpreisprojekt, darin ist mein Anteil ein Unterpunkt. Kunde und Ansprechpartner sitzen in Calgary, Kanada, die Projektsprache ist somit Englisch. Es finden wöchentliche Abstimmungsmeetings mit dem Kunden statt, ebenfalls anwesend ist dort der Projektleiter von Arsandis.

3.5.2 Sensus

Sensus ist seit der Gründung der Arsandis GmbH Kunde und nutzt seit über einem Jahrzehnt PTC Windchill PDMLink, welches aktuell von uns verwaltet wird. Es gibt schätzungsweise 400 aktive Windchill-Benutzer bei Sensus.

3.5.3 Pure Technologies Ltd

Pure Technologies Ltd ist der Auftraggeber und möchte nun ebenfalls das Windchill-System von Sensus nutzen. Dafür wird in dem momentan laufenden Projekt eine separate Organisationseinheit mit spezifisch angepassten Regeln und Funktionsweisen angelegt. Es wird voraussichtlich 58 aktive Windchill-Benutzer bei Pure Technologies Ltd geben.

3.5.4 Systeme

Es gibt 3 bestehende Systeme, diese brechen sich auf wie folgt:

- Produktivsystem

Das Produktivsystem wird von Sensus gehostet. Der Arsandis-Support, sowie alle Windchill-Benutzer von Sensus und Pure Technologies Ltd haben dort einen Zugang. Änderungen werden hier erst nach umfangreichen Tests auf dem Testsystem durchgeführt.

- Testsystem

Das Testsystem wird von Sensus gehostet. Der Arsandis-Support, das Arsandis-Testteam, sowie alle Windchill-Benutzer von Sensus und Pure Technologies Ltd haben dort einen Zugang. Das Testsystem wurde ebenfalls vor anderthalb Jahren geklont. Hier werden neue Änderungen am System durch die Kunden getestet.

- Entwicklungssystem

Das Entwicklungssystem wird von Arsandis gehostet (Hyper-V VM). Es ist nur Arsandismitarbeitern zugänglich und entstand als Klon des Produktivsystems vor anderthalb Jahren. Jegliche Entwicklungsarbeit startet auf dem Entwicklungssystem, sowie

die ersten Testrunden bevor Änderungen auf dem Testsystem nachgezogen werden. Meine Projektarbeit wird ausschließlich auf dem Entwicklungssystem stattfinden.

3.6 Ziel & Nutzen des Projektes

Ziel dieses Projektes ist die Umsetzung eines automatischen Nummerierungs-Schemas für CAD-Dateien und WTParts (Produktrepräsentationen), welches bei Bedarf von dem Benutzer umgangen werden kann.

Die automatische Nummerierung soll die Arbeit der Ingenieure durch Umgehen des langwierigen händischen Prozesses der Identifikation, Zuweisung und Reservierung von Nummern erleichtern.

3.7 Voraussetzungen für das Verständnis

3.7.1 Identifikationsattribute

Folgende Attribute werden von System und Benutzern für die Identifikation von CAD-Dateien und WTParts genutzt. Jedes dieser Attribute ist erforderlich und muss daher befüllt werden.

Dateiname

Der Dateiname ist nur bei CAD-Dateien vorhanden. Er muss systemweit einzigartig sein, da Referenzen zu abhängigen CAD-Dateien über den Dateinamen gebildet werden. Sollte der Dateiname nicht einzigartig sein, wird ein Hochladen oder Erstellen verwehrt. Derselbe Dateiname kann nicht gleichzeitig von Sensus und Pure Technologies Ltd genutzt werden.

Nummer

Die Nummer ist bei CAD-Dateien und WTParts vorhanden. Sie muss pro Organisationseinheit und Objekttyp einzigartig sein, da die Nummer eine organisationsweit eindeutige Kennung darstellt. Eine bestimmte Nummer kann von einer CAD-Datei und simultan von einem WTPart innerhalb der gleichen Organisationseinheit genutzt werden. Dieselbe Nummer kann gleichzeitig von Sensus und Pure Technologies Ltd genutzt werden.

Name

Der Name ist bei CAD-Dateien und WTParts vorhanden. Er muss nicht einzigartig sein und ist oft ein „sprechender“ (beschreibender) Name. Derselbe Name kann von Sensus und Pure Technologies Ltd beliebig oft verwendet werden.

3.7.2 Das WTPart-Konzept

Ein WTPart ist die Repräsentation eines Gegenstands oder Produktes als Ganzes. Es kann CAD-Daten referenzieren (Baugruppen, Komponenten und Zeichnungen), jegliche Dokumentation zu Tests, Normen, benötigten Freigabeinformation, Gebrauchsanleitungen und Weiteres beinhalten. Das WTPart bildet eine Auffangschale für alle relevanten Informationen zu dem abgebildeten Gegenstand oder Produkt.

In der Regel gibt es eine 1:1-Beziehung zwischen einem WTPart und einem CAD-Modell (Baugruppe oder Komponente). Diese wird „Owner-Link“ genannt. Dies führt dazu, dass das WTPart viele Informationen aus dem CAD-Modell automatisch übernimmt oder referenziert (Zeichnungen werden automatisch referenziert, bestimmte Attribute werden synchronisiert). Die WTPart-Struktur wird für die Erstellung von BOMs (bills of material bzw. Stücklisten) genutzt.

Hier als Beispiel eine CAD-Struktur mit ihrer zugehörigen WTPart-Struktur:

Abb. 1: CAD-Struktur















File Name	Number	Version
▲  iperl2030-main_cm.asm	IPERL2030-MAIN_CM.ASM	0.0
▲  iperl2030-housing_cm.asr	IPERL2030-HOUSING_CM.ASM	1.1
 iperl2030-housingmair	IPERL2030-HOUSINGMAIN_CM.PRT	0.0
 iperl2030-bottomlid_cr	IPERL2030-BOTTOMLID_CM.PRT	0.0
 iperl2030-newtoplid_cr	IPERL2030-NEWTOLID_CM.PRT	0.1
 iperl2030-connectorleft_cr	IPERL2030-CONNECTORLEFT_CM...	1.1
 iperl2030-connectorright_cr	IPERL2030-CONNECTORRIGHT_C...	0.0

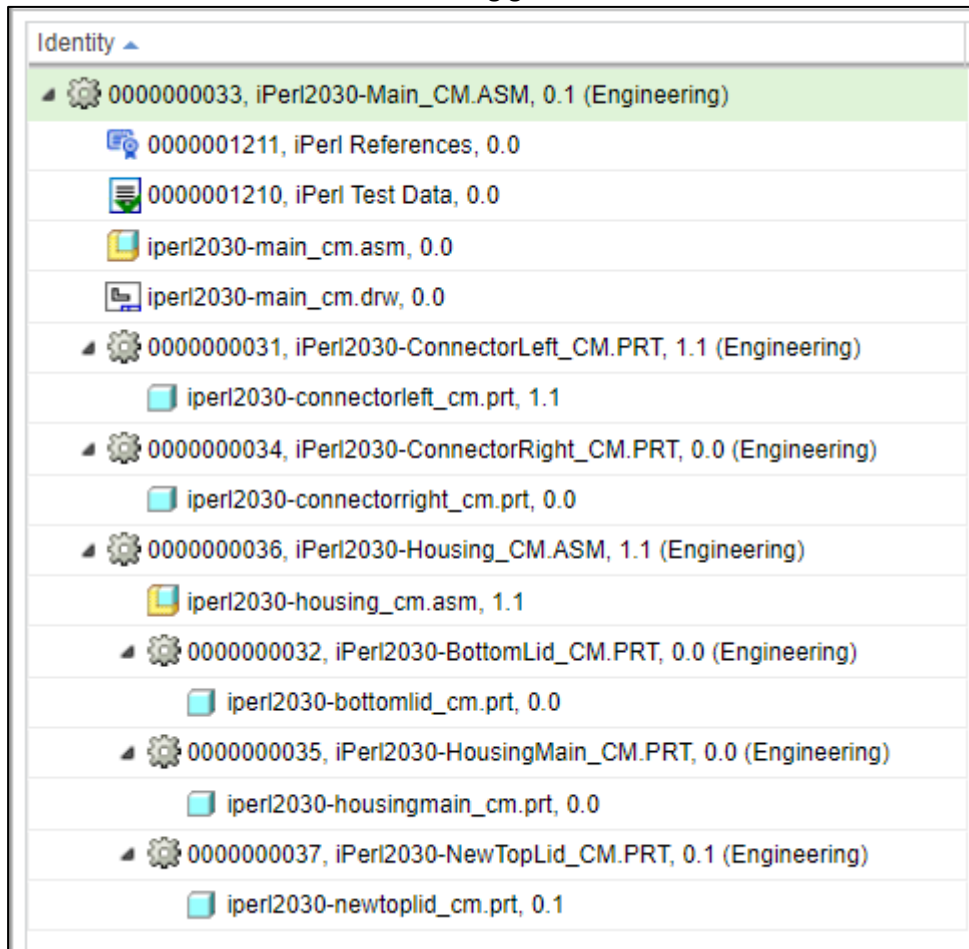
Abb. 2: WTPart-Struktur

Identity ▲
▲  0000000033, iPerl2030-Main_CM.ASM, 0.1 (Engineering)
 0000000031, iPerl2030-ConnectorLeft_CM.PRT, 1.1 (Engineering)
 0000000034, iPerl2030-ConnectorRight_CM.PRT, 0.0 (Engineering)
▲  0000000036, iPerl2030-Housing_CM.ASM, 1.1 (Engineering)
 0000000032, iPerl2030-BottomLid_CM.PRT, 0.0 (Engineering)
 0000000035, iPerl2030-HousingMain_CM.PRT, 0.0 (Engineering)
 0000000037, iPerl2030-NewTopLid_CM.PRT, 0.1 (Engineering)

Bei einem "Owner-Link" sollen das WTPart und das verbundene CAD-Modell dieselbe Nummer nutzen. Es gibt weitere Beziehungstypen zwischen WTParts und CAD-Modellen, diese sind jedoch "schwächer" (weniger bis gar keine Informationen werden geteilt) und die Nummern dürfen nicht synchron sein. Da sie für die Nummerierung keine Bedeutung haben, sind sie für diese Projektarbeit nicht relevant. Über Anzeigefilter können die Beziehungen zu anderen Objekten offengelegt werden.

Hier die WTPart-Struktur mit Anzeigefilter für alle Abhängigkeiten:

Abb. 3: WTPart-Struktur mit allen Abhängigkeiten



Ein WTPart kann auch Produkte ohne CAD-Modell darstellen (Kleber, Schmierstoff, Lötmittel und Weiteres).

3.7.3 Familientabellen

Bei einer Familientabelle gibt es ein "Parent"-Modell (Basismodell), welches die Grundform aller "Child"-Modelle (abgeleitete Modelle) vorgibt. In der Regel werden auf diesem Weg Bolzen, Wellen, Schrauben, Nägel und Ähnliches in einer Familie dargestellt. Meist unterscheidet sich nur eine Dimension zwischen dem Basismodell und allen abgeleiteten Modellen.

Durch eine Familientabelle wird die logische Zusammenfassung sehr ähnlicher Objekte unter einer einzelnen Basisnummer (+ individueller Suffix) möglich. Die Erstellung wird erheblich vereinfacht, da jedes einzelne Objekt nur gering von dem Basismodell abweicht, es beinhaltet also eine Art Kopiervorgang, bei welchem lediglich eine Abweichung modelliert wird.

4. Projektplanung

4.1 Projektphasen

Die Dokumentation wurde innerhalb jeder Phase laufend bearbeitet und hat somit keinen dedizierten Zeitraum.

4.1.1 Soll-Projektphasen

- Analysephase 04.10.21 – 05.10.21 (1 Tag)
 - o Ist-Analyse 2,5 Stunden
 - o Soll-Analyse 4,5 Stunden
 - o Erstellung Pflichtenheft 1 Stunde
- Entwurfsphase 05.10.21 – 07.10.21 (2 Tage)
 - o Recherchen zu Umsetzungsmöglichkeiten 8 Stunden
 - o Zusammenstellung des Gesamtkonzepts 6 Stunden
 - o Modellierung der Programmierungs-Grundlogik PAP 2 Stunden
- Implementierungsphase 08.10.21 – 13.10.21 (5 Tage)
 - o Generelle Systemkonfigurationen 8 Stunden
 - o Erstellung der Java-Klassen 32 Stunden
- Testphase 13.10.21 – 14.10.21 (1 Tag)
 - o Finaler Logiktest inkl. Bug Fixes 8 Stunden

4.1.2 Ist-Projektphasen

Abweichungen in lila.

- Analysephase 04.10.21 – 05.10.21 (< 1 Tag)
 - o Ist-Analyse 1 Stunde
 - o Soll-Analyse 4,5 Stunden
 - o Erstellung Pflichtenheft 1 Stunde
- Entwurfsphase 05.10.21 – 07.10.21 (> 2 Tage)
 - o Recherchen zu Umsetzungsmöglichkeiten 11 Stunden
 - o Zusammenstellung des Gesamtkonzepts 4 Stunden
 - o Modellierung der Programmierungs-Grundlogik PAP 2 Stunden
- Implementierungsphase 08.10.21 – 13.10.21 (5 Tage)
 - o Generelle Systemkonfigurationen 6 Stunden
 - o Erstellung der Java-Klassen 34 Stunden
- Testphase 13.10.21 – 14.10.21 (1 Tag)
 - o Finaler Logiktest inkl. Bug Fix 8 Stunden

4.1.3 Projektphasen-Vergleichs-Fazit

Die Ist-Analyse wurde wesentlich schneller abgeschlossen als angenommen. Ein Großteil der Arbeit während der Ausbildung des Prüflings wurde auf diesem System verrichtet(im Auftrag von Sensus), somit war der Ist-Zustand bereits gut bekannt.

Die gewonnene Zeit war nötig um nach Ablauf des 3ten Tages ein Gesamtkonzept mit modellierter Programmierlogik vorzeigen zu können. Insbesondere die für Recherchen benötigte Zeit wurde unterschätzt.

Die generellen Systemkonfigurationen waren einfach durchzuführen, welches 2 Stunden ersparte.

Für den Programmierteil dieses Projekts wurden daraufhin 2 Stunden zusätzlich genutzt. Durch ein Absehen von der Auslagerung der Logik in den beiden erstellten Klassen, war es durch einen Kopiervorgang möglich den Zeitplan einzuhalten (siehe „Number Generator“-Logik in 6.3.2). Die Endlösung ist nicht optimal, da es redundanten Code gibt, die Funktionalität ist jedoch in vollem Umfang gegeben.

Da trotz der Verschiebung einzelner Aktivitäten die groben Tagesabläufe und auch das Projektende eingehalten werden konnten, wird hier ein Erfolg verbucht.

4.1 Wirtschaftlichkeitsanalyse

Da die tatsächlichen Gesamtprojektkosten nicht bekannt gemacht wurden, rechnen wir hier mit einem mir empfohlenen Festpreis von 50.000€.

Der Anteil dieser Projektarbeit am Gesamtprojekt liegt bei geschätzten 10%. Gerechnet wird also mit einem Umsatz von 5000€ für die hier beschriebene Umsetzung.

4.1.1 Projektkosten

Wir gehen nun auf jeden einzelnen Kostenpunkt ein, um ein möglichst realistisches Bild zu zeichnen.

Auszubildenden-Vergütung

Hier haben wir ein Bruttoentgelt von 1080€.

Arbeitgeber SV-Anteile

Die Sozialabgaben für den Arbeitgeber belaufen sich auf 218,43€, exclusive Unfallversicherung.

Für die Unfallversicherung liegen mir keine Zahlen vor, eine Internetrecherche bescherte mir ebenfalls keine Klarheit. Laut unserer Personalabteilung soll ich einen Anteil von 20€ monatlich für meine Rechnung nutzen.

Abstimmungskosten

Es wurden mit dem Projektleiter seitens Arsandis einige interne Abstimmungsgespräche zum Projektfortschritt durchgeführt. Mit den Gesprächen sollte sichergestellt werden, dass die Qualitätsanforderungen eingehalten werden, die Umsetzung nach „Best Practices“ erfolgt und die Lösung sich gut in den Gesamtentwurf des Kundensystems einbettet. Weiterhin nahm der Projektleiter – wie in anderen Projektanteilen auch – an Meetings mit dem Kunden teil.

Direkte Zahlen sind für mich nicht einsehbar. Mir wurde empfohlen, mit einer festen Rate von 100€ pro Stunde zu rechnen.

Gemeinkosten

Wir verwenden firmenintern keinen Gemeinkostenfaktor. Abgesehen davon, müssen wir unsere Gemeinkosten trotzdem decken. Daher werde ich eine weitere Annahme treffen, um hier eine übliche Wirtschaftlichkeitsanalyse durchführen zu können. Die tatsächlichen Zahlen sind für mich nicht einsehbar.

- Marketingkosten: Abgedeckt durch eine Auszubildende = 7,5%
- Vertrieb: Abgedeckt durch einen dualen Studenten = 7,5%
- Energiekosten: Heizung, Strom (Monitore, Clients, Server etc.) = 10%
- Mietkosten: Büroräume = 15%
- Ausrüstungskosten: Clients, Monitore, Server, Peripherie, Lizenzen = 15%

Damit kommen wir auf einen Gemeinkostenzuschlag von 55%.

Gesamtkosten

Berechnet werden hier die Kosten eines Arbeitstages des Prüflings durch eine Ermittlung der Kosten über ein Jahr und das Runterbrechen auf einen Tag aus 220 und eine anschließende Hochrechnung auf 9 Arbeitstage. Letzten Endes werden auch die Kosten der Abstimmungsgespräche seitens des Projektleiters aufgerechnet.

Die Kosten der erbrachten Arbeitstage des Prüflings errechnen sich wie folgt:
(Bruttoentgelt eines Monats + SV-Abgaben des Arbeitgebers eines Monats) * 12 Monate eines Jahres
* 1,55 Gemeinkostenzuschlag / 220 Tage in einem Jahr * 9 erbrachte Arbeitstage
(1080€ + 238,43€) * 12 * 1,55 / 220 * 9 = 1003,20€

Die Kosten der Abstimmungsgespräche errechnen sich wie folgt:
fester Stundensatz * 6 erbrachte Stunden * 1,55 Gemeinkostenzuschlag
 $100\text{€} * 6 * 1,55 = 933\text{€}$

Die Gesamtkosten errechnen sich wie folgt:
Kosten der Prüflingsarbeitstage + Kosten der Abstimmungsgespräche
 $1003,20\text{€} + 933\text{€} = 1936\text{€}$

4.1.2 Amortisation

Nun errechnen wir den Gewinn durch eine simple Gegenüberstellung von Umsatz und Kosten:
Umsatz - Gesamtkosten = Gewinn
 $5000\text{€} - 1936,20\text{€} = 3063,8\text{€}$

Im Anschluss eine Rechtfertigung für den hohen Gewinn:

Da der Prüfling fast fertig ausgebildet ist und in einem Bereich eingesetzt wurde, für welchen er bereits einen Großteil des benötigten Wissens besaß, fiel der Gewinn sehr hoch aus. Hätte sich das Projekt nur ein paar Monate später ereignet, wäre dieser durch die stark gestiegenen Lohnkosten wesentlich geringer ausgefallen.

5. Definition der Anforderungen

5.1 Lastenheft

Ein Lastenheft im engeren Sinne wurde nicht bereitgestellt. Vielmehr wurden die Anforderungen in Zusammenarbeit mit dem Kunden entwickelt und im Anschluß an das Gespräch wie folgt dokumentiert:

For auto-naming and numbering, the following rules shall apply:

- File name: Mapped from the object's file name
- Number: Auto-generated, starting at 1000000, increments by 1
 - o Sequence is shared with WTParts
 - o Family Parents shall have a "-XX" suffix
 - o Family Children suffix is "-01" instead, increments by 1 per additional child
- Name: Mapped from the Creo attribute "Common Name"
 - o Name shall be limited to 60 characters for the part type "mech – purchased" or "mech – assy" and 60 characters for all other part types.
 - o If different character limits are not feasible, a global limit of 60 characters is acceptable

5.2 Ausgangszustand

Zum jetzigen Zeitpunkt werden Nummern für neue CAD-Dateien und WTParts im System nach dem OOTB-Schema (out-of-the-box) vergeben.

Es gibt 2 verschiedene Arten CAD-Dateien in Windchill zu erfassen:

- Hochladen über ein CAD-Programm
- Assoziation während der händischen Erstellung eines WTParts über das Windchill UI.

Es gibt 3 verschiedene Arten WTParts zu erstellen

- Erstellung durch eine Assoziation in Windchill vorhandener CAD-Dateien
- Erstellung durch eine Assoziation während des Hochladens von CAD-Dateien zu Windchill
- Manuelle Erstellung über das Windchill UI.

Ist- und Soll-Analyse werden diese 5 Wege der Datenanlage betrachten.

5.3 Ist-Analyse

5.3.1 Hochgeladene CAD-Dateien

Der reguläre Prozess des Erfassens einer CAD-Datei in Windchill läuft über eine verbundene CAD-Anwendung. Die CAD-Datei wird in der CAD-Anwendung erstellt und dann durch den Benutzer über die "Checkin"-Funktion in die von Windchill genutzte Datenbank hochgeladen.

Dateiname

Der Dateiname wird bei der Anlage der Datei von dem Benutzer festgelegt, welches in der genutzten CAD-Anwendung stattfindet.

Nummer

Die Nummer wird während des Hochladens der Datei festgelegt. Hierzu wird lediglich der Dateiname (inklusive Dateiendung) als Nummer übernommen.

Name

Der Name wird bei der Anlage der Datei von dem Benutzer festgelegt, welches in der genutzten CAD-Anwendung stattfindet.

5.3.2 Assoziierte CAD-Dateien

Die Alternative für die Erfassung einer CAD-Datei ist die händische Erstellung in Kombination mit einem WTPart über das Windchill UI. Hier hat der Benutzer während der Erstellung eines WTParts die Möglichkeit nach Angabe aller benötigten Informationen eine neue CAD-Datei zu assoziieren. Für die Erstellung der CAD-Datei sind weitere Eingaben des Benutzers erforderlich.

Dateiname

Der Dateiname wird durch den Benutzer über eine Eingabe definiert. Die Möglichkeit den Dateinamen aus der Nummer der CAD-Datei zu kopieren, besteht durch die Nutzung einer "Checkbox".

Nummer

Die Nummer wird durch den Benutzer über eine Eingabe definiert. Die Möglichkeit die Nummer der CAD-Datei von der Nummer des WTParts zu kopieren, besteht durch die Nutzung einer "Checkbox".

Name

Der Name wird automatisch mit dem Namen des WTParts vorbefüllt (Dateiendung wird angehängen), kann jedoch durch den Benutzer überschrieben werden.

5.3.3 Assoziation von WTParts während des Hochladens von CAD-Dateien

Der reguläre Prozess für die Erstellung eines WTParts ist die Assoziation zu einem CAD-Modell. Dieser Vorgang kann während des Hochladens einer CAD-Datei aus einem CAD-Programm zu Windchill stattfinden (durch Nutzung einer "Checkbox"). Hierbei ist keine Eingabe des Benutzers möglich, alle Attribute werden automatisch befüllt.

Nummer

Die Nummer wird von der CAD-Datei übernommen.

Name

Der Name wird von der CAD-Datei übernommen.

5.3.4 Assoziation von WTParts zu CAD-Dateien in Windchill

Die Erstellung durch eine Assoziation zu in Windchill vorhandenen CAD-Dateien ist ebenfalls möglich. Hierbei hat der Benutzer die Möglichkeit die vorbefüllten Werte zu überschreiben, bevor das WTPart angelegt wird.

Nummer

Die Nummer wird durch das System mit der Nummer der CAD-Datei vorbefüllt, kann jedoch durch den Benutzer überschrieben werden.

Name

Der Name wird durch das System mit dem Namen der CAD-Datei vorbefüllt, kann jedoch durch den Benutzer überschrieben werden.

5.3.5 Händisch erstellte WTParts

Die Erstellung über einen Knopf des Windchill UIs ist die letzte Alternative.

Nummer

Die Nummer wird durch das System vorbefüllt, kann jedoch durch den Benutzer überschrieben werden. Hierbei handelt es sich um das OOTB-Nummerierung-Schema für WTParts. Dieses ist nicht mit dem Nummerierung-Schema von Pure Technologies Ltd konform.

Name

Der Name muss durch den Benutzer gewählt werden.

5.4 Pflichtenheft

Das von Pure Technologies Ltd genutzte Schema sieht die individuelle Nummerierung und die Nutzung von Familientabellen vor.

Grundnummerierung:

- Die Nummer besteht aus Sieben Ziffern
- Der Startwert liegt bei 1.000.000
- Die Nummer wird um 1 inkrementiert
- Dateiendungen werden nicht in die Nummer übernommen
- Umgehen der Nummerierung soll möglich sein (für evtl. spätere Nummerierung)

Familientabellen:

- Basieren auf einem "Parent"-Modell
- "Parent"-Modell hat als Nummernsuffix "-XX"
- "Child"-Modelle mit identischer Grundnummer, jedoch anderem Nummernsuffix
- "Child"-Modell Nummernsuffix startet bei "-01"
- "Child"-Modell Nummernsuffix wird um 1 inkrementiert
- "Child"-Modell kann Nummern mit einem maximalen Suffix "-999" annehmen

5.5 Soll-Analyse

Es gilt die Nummerierung in einer Weise anzupassen, welche bei jedem dieser 5 Erstellungswege für CAD-Dateien und WTParts automatisch die korrekte Nummer zuweist, jedoch dem Benutzer weiterhin die manuelle Nummerierung ermöglicht, falls erwünscht.

Genannt werden nur Punkte, welche einer Anpassung bedürfen, Änderungen sind markiert (in **lila**). Sonstige Punkte bekommen lediglich einen Verweis auf ihre Abhängigkeiten.

5.5.1 Hochgeladene CAD-Dateien

Der reguläre Prozess des Erfassens einer CAD-Datei in Windchill läuft über eine verbundene CAD-Anwendung. Die CAD-Datei wird in der CAD-Anwendung erstellt und dann durch den Benutzer über die "Checkin"-Funktion in die von Windchill genutzte Datenbank hochgeladen.

Dateiname

Der Dateiname wird bei der Anlage der Datei von dem Benutzer festgelegt, welches in der genutzten CAD-Anwendung stattfindet.

Nummer

Die Nummer wird bei dem "Checkin" der Datei festgelegt. **Hier wird eine „Parent“- oder „Child“-Nummer, die nächste freie Nummer der Sequenz oder bei Bedarf, eine aus dem Dateinamen abgeleitete Nummer vergeben.**

Name

Der Name wird bei der Anlage der Datei von dem Benutzer festgelegt, welches in der genutzten CAD-Anwendung stattfindet.

5.5.2 Assoziierte CAD-Dateien

Nummernübernahme ist abhängig von korrekter Nummerierung des assoziierten WTParts (Umsetzung erfolgt durch: 4.5.5).

5.5.3 Assoziation von WTParts während des Hochladens von CAD-Dateien

Nummernübernahme ist abhängig von korrekter Nummerierung der assoziierten CAD-Datei (Umsetzung erfolgt durch: 4.5.1).

5.5.4 Assoziation von WTParts zu CAD-Dateien in Windchill

Nummernübernahme ist abhängig von korrekter Nummerierung der assoziierten CAD-Datei (Umsetzung erfolgt durch: 4.5.1).

5.5.5 Händisch erstellte WTParts

Die Erstellung über einen Knopf des Windchill UI ist die letzte Alternative.

Nummer

Die Nummer wird durch das System vorbefüllt, kann jedoch durch den Benutzer überschrieben werden. **Hier wird die nächste freie Nummer der Sequenz oder eine „Parent“- oder „Child“-Nummer vergeben.**

Name

Der Name muss durch den Benutzer gewählt werden.

6. Entwurfsphase

Hier wird ein Entwurf für alle benötigten Änderungen erstellt, es muss an verschiedenen Stellen im System angesetzt werden, um die Anforderungen gänzlich abzudecken. Die Änderungen werden grob nach Konfigurationsart unterteilt (Systemkonfigurationen und Java-Programmierung).

Nachdem ein Grundverständnis für die Zielplattform etabliert wurde, werden die Änderungen im einzelnen erläutert.

6.1 Zielplattform

Bei der Zielplattform handelt es sich um PTC Windchill PDMLink, eine web-basierte Anwendung. Benutzer können sich ganz einfach über einen Browser ihrer Wahl, je nach Konfiguration über das Intranet oder das Internet einloggen. Windchill ist eine Java-Applikation, darum werden tiefer gehende Anpassungen durch Java-Code erfolgen.

Die Grundbestandteile von Windchill sind wie folgt:

6.1.1 Windchill PDMLink

Dies ist das Windchill "Standard-Paket". Es wird als Basis für andere Windchill-Module genutzt und bietet bereits zahlreiche Funktionen, welche für die Produktentwicklung essenziell sind (z.B. change management, promotion processes, versioning, access control).

Windchill 11.1 M020 CPS11 ist im Einsatz.

6.1.2 LDAP

Sollte kein Bestehendes vorhanden sein (AD oder sonstiges) muss bei der Installation von Windchill eine entsprechende Anwendung erstellt werden um Benutzer- und Gruppeninformationen, sowie weitere bestimmte Metadaten aufzunehmen.

OpenDJ 3.0.0 ist im Einsatz.

6.1.3 Datenbank

Hier werden sämtliche Informationen des Systems gespeichert, bis auf hochgeladene Dateien. Während die Metadaten (z.B. in Windchill vergebene Attribute) sich in der Datenbank befinden, werden die tatsächlichen Dateiinhalte außerhalb der Datenbank, lokal auf dem Applikations-Server platziert.

Microsoft SQL Server 2016 ist im Einsatz.

6.1.4 Webserver

Der Webserver ist eine weitere benötigte Komponente von Windchill.

Apache 2.4 ist im Einsatz.

6.1.5 Java (JRE)

Die letzte erforderliche Komponente ist eine JRE. Sollte während der Installation von Windchill noch keine kompatible Version vorhanden sein, kann ein im PSI (PTC Solution Installer) gebündeltes JDK 8 installiert werden.

Java Development Kit 1.8.0 Update 202 ist im Einsatz.

6.2 Entwurf für benötigte Systemkonfigurationen

6.2.1 Datenbank-Sequenz

Diese Sequenz wird der Träger der neuen Nummerierung. Ein Template für die Erstellung einer Sequenz stellt PTC online bereit.

6.2.2 Preferences

Es werden Preferences auf Ebene der neuen Organisation angepasst.

Es wurden 3 benötigte Preferences identifiziert. Diese können von einem Administrator innerhalb kürzester Zeit angepasst werden. Der größte Aufwand ist hier die Identifikation der benötigten Preferences, über die von PTC bereitgestellte Dokumentation. Die Preferences werden gesperrt, um das Überschreiben der Preference durch Benutzer für ihren eigenen Kontext zu verhindern. Damit wird eine konsistente Nummerngenerierung sichergestellt.

6.2.3 OIR (Object Initialization Rule)

OIRs beeinflussen das Verhalten eines Objekttyps bei der Erstellung innerhalb des Systems. Eine OIR wird in Form einer XML-Datei auf dem Server hinterlegt. Es werden OIRs spezifisch für die neue Organisationseinheit angelegt. Ein OIR bezieht sich immer nur auf den angegebenen Objekttyp selbst und auf von diesem Erbende.

Eine Anpassung für WTParts ist erforderlich, um bei der Erstellung im System die Nummer korrekt zu befüllen. Es werden ebenfalls während der Erstellung angegebene Attribute innerhalb der OIR weitergereicht.

Die tatsächliche Logik der Nummernvergabe ist hierbei nicht in der OIR enthalten, stattdessen wird eine Java-Klasse aufgerufen, welche die Nummer als Rückgabewert liefert. Nach der Rückgabe wird die Nummer durch die OIR zugewiesen.

6.2.4 Attribute

Zusätzliche Attribute werden benötigt, um Child- oder Parent-Nummern korrekt zu vergeben und die Autonummerierung während des Hochladens einer CAD-Datei zu umgehen. Sie sind nur relevant während der Anlage eines neuen Objekts (CAD-Datei, WTPart), da zu diesem Zeitpunkt die Objektnummer vergeben wird.

Außerhalb der Erstellung sollen diese Attribute nach Möglichkeit nicht mehr für den Endanwender sichtbar sein, um eventuelle Verwirrungen zu vermeiden. Diese Attribute werden nach der Erstellung des Objekts nicht dazu in der Lage sein eine Veränderung der Nummer zu bewirken.

Die definierten Attribute werden ebenfalls als Parameter in den CAD-Templates der neuen Organisationseinheit angelegt, auf diese kann dann während des Hochladens der Datei zugegriffen werden, um eine korrekte Nummer zu generieren. Die Erstellung der CAD-Templates wird von einem Xylem-Mitarbeiter übernommen. Für WTParts sind nicht alle Attribute benötigt.

6.3 Entwurf für benötigte Java-Programm Logik

6.3.1 Benötigte Klassen

Es werden 2 neue Java-Klassen benötigt:

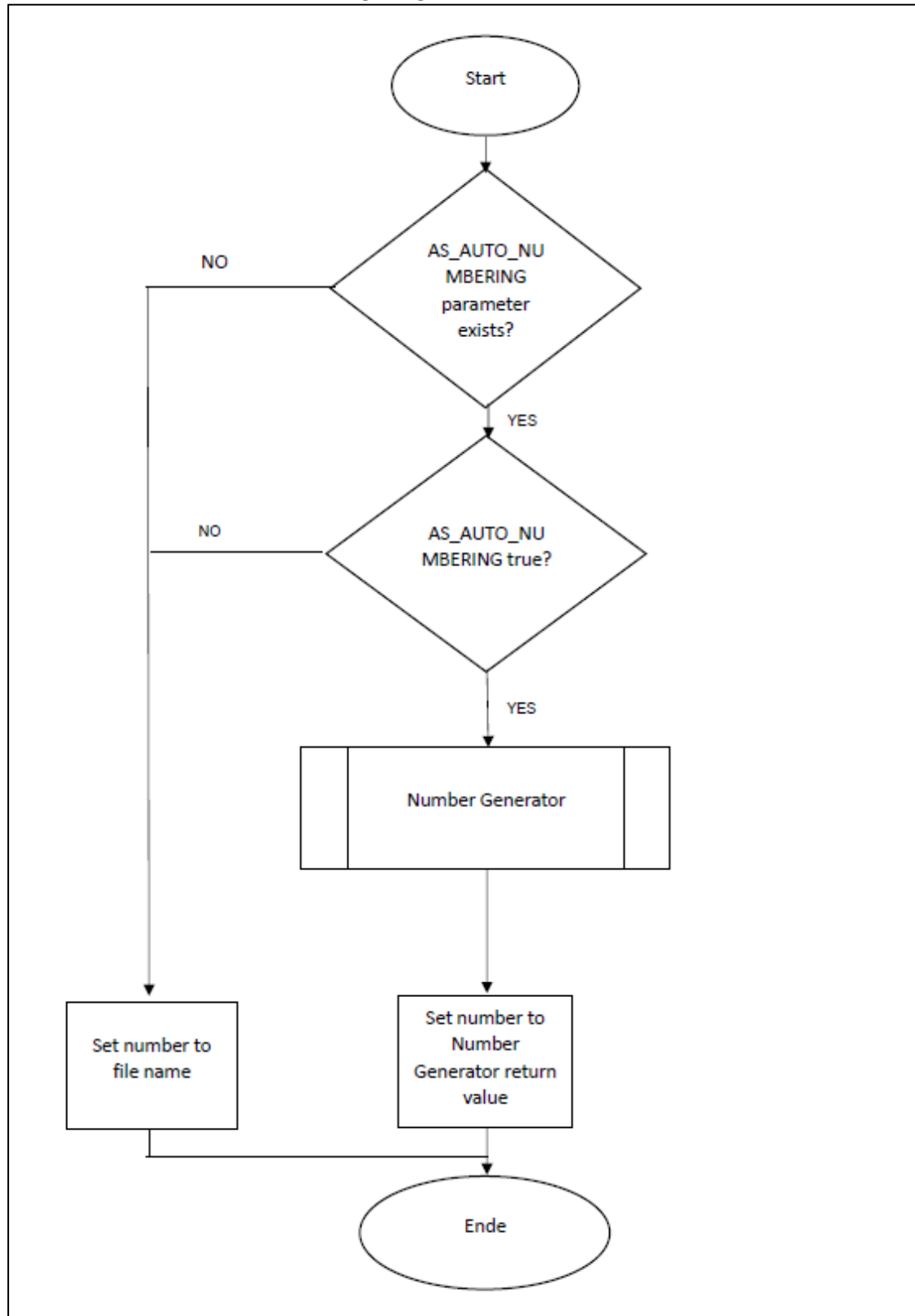
- ASPartNumberingRule, eine Klasse für die händische Erstellung von WTParts im UI von Windchill (Aufruf erfolgt über die OIR)
- ASEPMDocumentNamingDelegate, eine Klasse für die Nummerierung von CAD-Dateien während des Uploads aus einem CAD-Programm zu Windchill (Implementierung eines

Interface von PTC mit Überschreibung einer Methode, welche während des Uploads auf CAD-Dateien zugreifen und Nummer sowie Name beeinflussen kann)

6.3.2 Grundlogik in PAP-Form

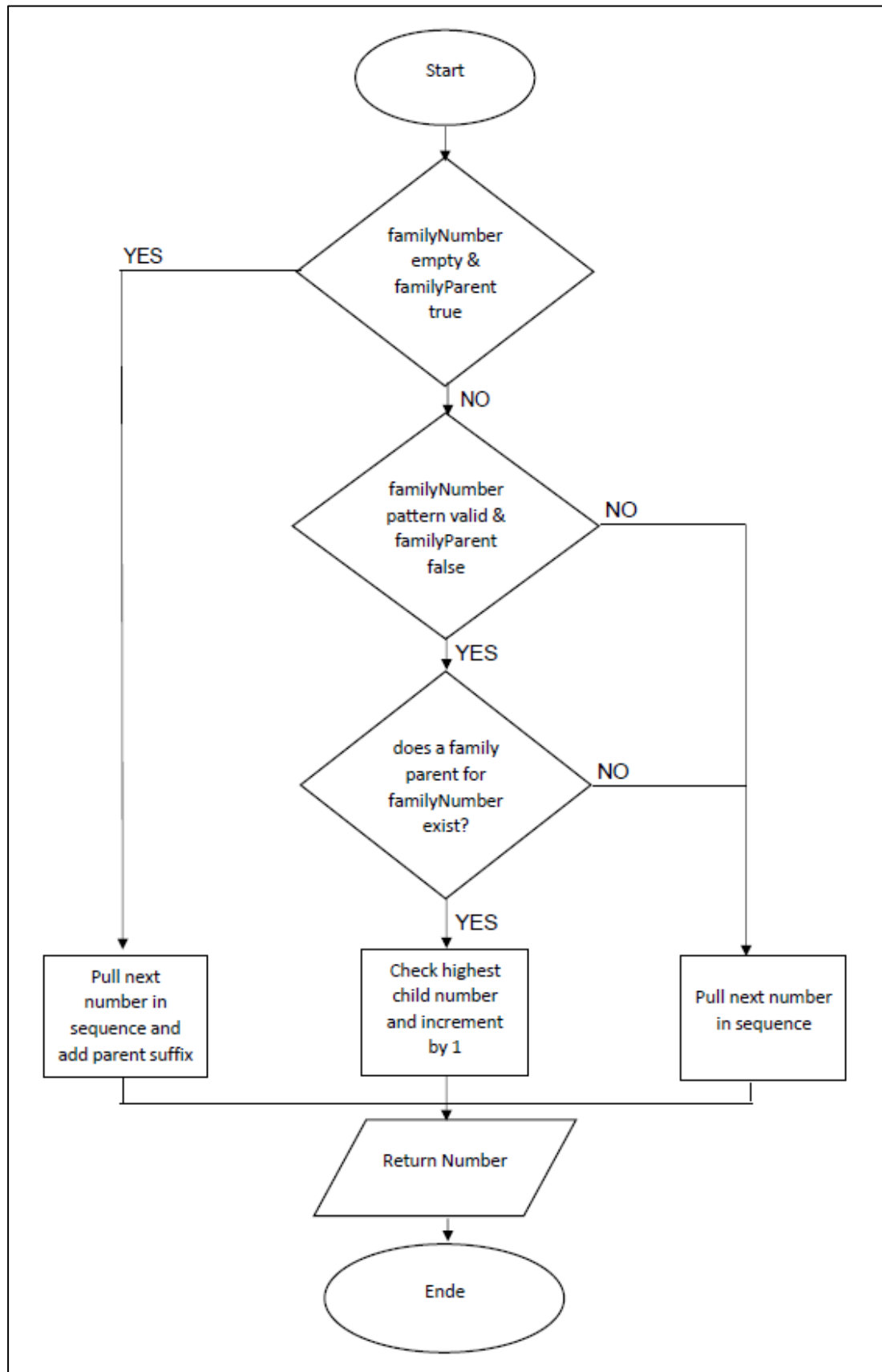
Logik des ASEPMDocumentNamingDelegate:

Abb. 4: PAP ASEPMDocumentNamingDelegate



Logik des Number Generators, findet Verwendung in der Klasse ASPartNumberingRule (Aufruf durch OIR) und der Klasse ASEPMDocumentNamingDelegate:

Abb. 5: PAP Number Generator



7. Implementierungsphase

Jegliche hier gelistete Implementierungsarbeit wurde durch den Prüfling vorgenommen, es sei denn es wird explizit darauf hingewiesen, dass ein Arbeitsschritt nicht durch den Prüfling vorgenommen wurde.

7.1 Erstellung einer Datenbank-Sequenz

Hier wird das online von PTC bereitgestellte Template genutzt, das Template befindet sich im Anhang 12.1. Angepasst wurden lediglich Startwert und Name der Sequenz.

Über einen Kommandozeilenbefehl wird das Template zur Erstellung einer Sequenz verwendet.

```
D:\ptc\Windchill\Windchill\db\execute_sql_script.bat create_ASPartSeq_sequence.sql <dbuser> <password>
```

7.2 Anpassung der Preferences

Über das UI von Windchill können die als benötigt identifizierten Preferences gesetzt und gelockt werden.

Operation > Upload Operation > Upload > Drop File Extension From Name = Yes

Operation > Upload Operation > Upload > Drop File Extension From Number = Yes

Operation > New CAD document > Set CAD Document Number Same As Part Number = Yes

Abb. 6: Preference Management

Name	Value	Description	Internal Name
Add to Baseline		Add to Baseline operation preferences	ADDTOBASLINE_CATEGORY
Advanced Assembly Editor		Advanced Assembly Editor Preferences	ATE_CATEGORY_NAME
Arbortext		Arbortext Preferences	ARBORTEXT_CATEGORY_NAME
Associativity		Contains the preferences that are specific to the associativity area.	Associativity
Attachments			ATTACHMENT_CATEGORY
Attribute Handling			ATTRIBUTE_HANDLING_CATEGORY
Business Rules			BUSINESS_RULES_CATEGORY
Change Management		Change Management preferences	CHANGE_MANAGEMENT_CATEGORY
Configurable Links		Configurable Links preferences	CONFIG_LINKS_CATEGORY_NAME
Create and Edit			CREATE_EDIT_HANDLING_CATEGORY
Delete		Delete preferences	DELETE_OBJECT_CATEGORY
Display			DISPLAY_CATEGORY
Display Related Manufacturing Objects		Display Related Manufacturing Objects report preferences	DISPLAYRELATEDMANUFACTURINGITEMS_CATEGORY
Documents			DOCUMENTS_CATEGORY
Effectivity Management			EFFECTIVITY_MANAGEMENT_CATEGORY
EPM Service Preferences		EPM service preferences	EPM
ext		This category contains custom preferences.	EXT_CUSTOM_PREF_ROOT_CATEGORY
General Collection		General collection preferences	GENERAL_COLLECTION
Integral Operations		Integral Operations action preferences. Contains preferences like Move, Add to Project and Send to PDM collector.	INTEGRAL_OPERATIONS_CATEGORY
Managed Collections			MANAGED_COLLECTION_CATEGORY
Meeting			MEETING
Notification			NOTIFICATION_CATEGORY
Operation		Category for Operations preferences	UWGM_OPERATION
Options and Variants		Preferences Group for Options and Variants	OV_PREF_GROUP
Packages			WORK_PACKAGE_CATEGORY
Part Management			PART_MANAGEMENT
Project Planning			PROJECT_PLANNING

7.3 Anpassung der OIR

Die OIR für WTParts innerhalb der neuen Organisationseinheit wurde angepasst, um die Java-Klasse ASPartNumberingRule für die Nummerngenerierung zu nutzen (es werden ausserdem einige Attribute des Objekts übergeben, benötigt für die Nummerierungslogik der Java-Klasse). Der Inhalt der Datei befindet sich im Anhang 12.2.

Hier wurden folgende Zeilen durch den Prüfling angepasst:

Das value für "algorithm" übergibt den Namen der Klasse, welche die Logik für die Nummerngenerierung enthält. Der Inhalt der „Arg“-Tags gibt die Sequenz weiter an die Klasse. Die beiden Attribute darunter definieren weitere Übergabewerte für die Klasse. Da sie auf dem Objekt WTPart angelegt wurden, können Ihre Werte hier übergeben werden (Identifikation über internen Namen des Attributs).

```
<!-- set the number to a generated number -->
<AttrValue id="number" algorithm="ext.as.part.ASPartNumberingRule">
  <Arg>{GEN:wt.enterprise.SequenceGenerator:ASPartSeq:7:0}</Arg>
  <Attr id="IBA|com.as.FAMILY_NUMBER"/>
  <Attr id="IBA|com.as.FAMILY_PARENT"/>
</AttrValue>
```

Über die attribute constraints, wird festgelegt, dass der Server bei Erstellung nach den in der OIR genannten Regeln eine Nummer für ein neues Objekt vorbefüllt. Ein „immutable“-constraint ist nicht vorhanden, damit Benutzern die Überschreibung des vorbefüllten Wertes möglich ist.

```
<!-- specify AttrConstraint tag -->
<AttrConstraint id="number" algorithm="com.ptc.core.rule.server.impl.GatherAttributeConstraints">
  <Value algorithm="com.ptc.core.rule.server.impl.GetServerAssignedConstraint"/>
</AttrConstraint>
```

Hinterlegen der XML-Datei erfolgt über das Windchill UI:

Abb. 7: Object Initialization Rule Administration

PLM-Dev barry_al

Organizations > Assessment Services

Object Initialization Rules

Delete New Object Initialization Rule Download Composite Rule

Name	Type	Context	Enabled
Action Item	Action Item	Site	Yes
Agenda	Agenda	Site	Yes
Agreement	Agreement	Site	Yes
AS CAD Document	AS CAD Document	Assessment Services	Yes
AS Change Notice	Change Notice Assessment Services	Assessment Services	Yes
AS Change Request	Change Request Assessment Services	Assessment Services	Yes
AS Change Task	Change Task Assessment Services	Assessment Services	Yes
AS Document	AS Document	Assessment Services	Yes
AS Part	AS Part	Assessment Services	Yes
AS Problem Report	Problem Report Assessment Services	Assessment Services	Yes
AS Promotion Request	Promotion Request	Assessment Services	Yes
CAPA Action Plan	CAPA Plan	Site	Yes
CAPA Activity	CAPA Activity	Site	Yes
CAPA Investigation	CAPA Investigation	Site	Yes
CAPA Request	CAPA Request	Site	Yes
Change Activity	Change Task	Site	Yes
Change Directive	Change Directive	Site	Yes
Change Issue	Problem Report	Site	Yes
Change Notice	Change Notice	Site	Yes
Change Notice Sensus EM	Change Notice Sensus EMEAAP	Site	Yes
Change Notice Sensus NA	Change Notice Sensus NA	Site	Yes
Change Proposal	Change Proposal	Site	Yes
Change Request	Change Request	Site	Yes
Change Request Sensus EM	Change Request Sensus EMEAAP	Site	Yes
Change Request Sensus NA	Change Request Sensus NA	Site	Yes
Change Review	Review	Site	Yes
Change Task Sensus	Change Task: General	Site	Yes

(0 objects selected)

7.4 Erstellung der Attribute

Die Attribute werden über das Windchill UI als Administrator hinzugefügt. Es werden Attribute für die Objekttypen WTPart und EPMDocument (Objektyp für CAD-Dateien) hinzugefügt.

Benötigte Attribute sind:

- FAMILY_PARENT (CAD-Dokument, WTPart)
- FAMILY_NUMBER (CAD-Dokument, WTPart)
- AS_AUTO_NUMBERING (CAD-Dokument)

Das Hinzufügen neuer Attribute ist über das Windchill UI möglich.

Attribute für CAD-Dokumente:

Abb. 8: Type and Attribute Management – AS CAD Document

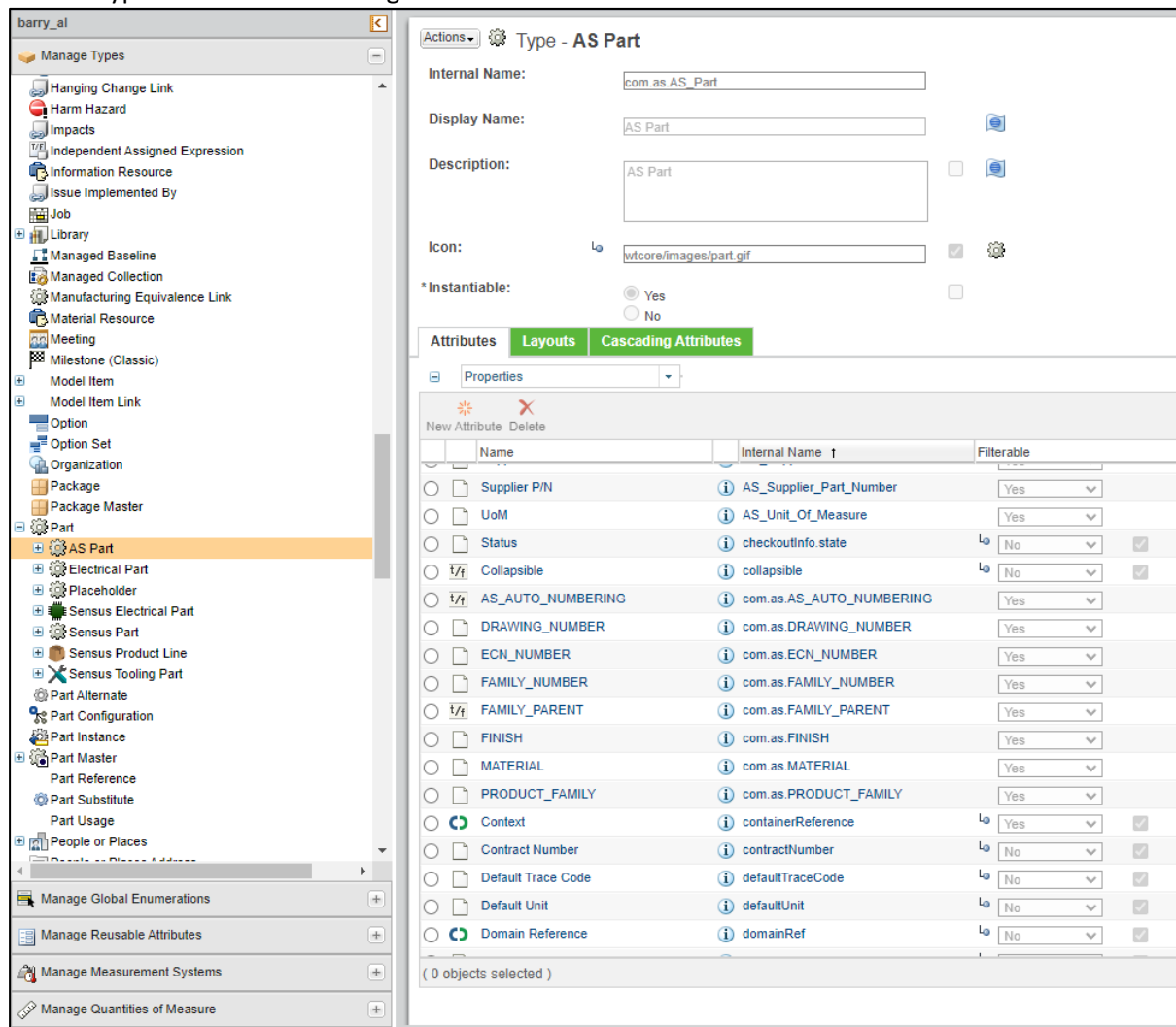
The screenshot shows the Windchill UI for managing attributes. On the left, a tree view under 'Manage Types' shows 'AS CAD Document' selected. The right pane, titled 'Type - AS CAD Document', has tabs for 'Attributes' and 'Layouts'. The 'Attributes' tab is active, showing a table of attributes with columns: Name, Internal Name, and Filterable. The table lists various attributes, with 'AS_AUTO_NUMBERING' selected.

Name	Internal Name	Filterable
boxExtents.Bz	boxExtents.Bz	
File Name	CADName	Yes
Status	checkoutInfo.state	
Collapsible	collapsible	
APPROVED_BY_CHANGED	com.as.APPROVED_BY_CHANGED	No
APPROVED_BY_CREATED	com.as.APPROVED_BY_CREATED	No
AS_AUTO_NUMBERING	com.as.AS_AUTO_NUMBERING	No
AS_CAD_FILE	com.as.AS_CAD_FILE	No
CHANGED_BY	com.as.CHANGED_BY	No
CREATED_BY	com.as.CREATED_BY	No
DRAWING_NUMBER	com.as.DRAWING_NUMBER	No
ECN_NUMBER	com.as.ECN_NUMBER	No
FAMILY_NUMBER	com.as.FAMILY_NUMBER	No
FAMILY_PARENT	com.as.FAMILY_PARENT	No
FINISH	com.as.FINISH	No
MASS_DESIGNATED	com.as.MASS_DESIGNATED	No
MATERIAL	com.as.MATERIAL	No
PRODUCT_FAMILY	com.as.PRODUCT_FAMILY	No

(0 objects selected)

Attribute für WTParts:

Abb. 9: Type and Attribute Management – AS PART



7.5 Erstellung der Java-Klassen

Es wurden 2 Java-Klassen erstellt:

- ASEPMDocumentNamingDelegate befindet sich im Anhang 12.3
- ASPartNumberingRule befindet sich im Anhang 12.3

Der Code wird über eine IDE kompiliert und dann in die Codebase verschoben. Bei der Codebase handelt es sich um einen Ordner im Windchill-Installationsverzeichnis, welcher den gesamten kompilierten Java-Code beinhaltet. Werden hier Class-Dateien eingefügt, kann Windchill sie finden und bei Bedarf (Aufruf) nutzen.

8. Qualitätssicherung

Die QS enthält lediglich laufende Tests während der Erstellung der Java-Klassen und einen finalen Logiktest, welcher durch den Prüfling vorgenommen wurde.

Da die Implementierung nicht innerhalb des Projektbearbeitungszeitraums auf dem internen Testsystem des Kunden stattfindet, wird keine Kundentestphase dokumentiert. Diese wird zu einem späteren Zeitpunkt jedoch stattfinden.

8.1 Testvorgang

Hierzu wird regelmäßig nach einer Änderung der Programmierung testweise ein Objekt angelegt (WTPart oder hochgeladene CAD-Datei) um die Korrektheit der Nummerierung zu überprüfen.

8.2 Finaler Logiktest

Hier wird gegen einen Testplan getestet, welcher die Punkte aus dem Pflichtenheft mit zusätzlichen Variationen, sowie weitere Testfälle zu Familientabellen aufgreift. Der Testplan befindet sich im Anhang 12.5.

Der finale Logiktest identifizierte einen Fehler, welcher das Hochladen mehrerer CAD-Dateien als „Child“ derselben Familie verhinderte.

Ein Hochladevorgang ist eine Transaktion, somit wird in der Datenbank erst geschrieben, sobald alle Teile der Transaktion erfolgreich abgeschlossen sind. Da der Code bei Anlage jedes „Child“-Objektes in der Datenbank die nächste freie Nummer ermittelte, kam es hier zu Fehlern bei der Vergabe von „Child“-Nummern des selben „Parents“.

```
//Identifying the highest child number known to the database
int familyChildNumber = 1;
while (qrChildren.hasMoreElements()){
    String currentEPMNumber = ((EPMDocument)qrChildren.nextElement()).getNumber();
    if (Pattern.matches(FAMILY_CHILD_REGEX, currentEPMNumber)){
        int currentChildNumber = Integer.parseInt(currentEPMNumber.substring(8));
        if (currentChildNumber >= familyChildNumber){
            familyChildNumber = currentChildNumber + 1;
        }
    }
}
```

Jedem „Child“ wurde dieselbe Nummer zugewiesen, da keine der neuen Nummern vor Ende der Transaktion in der Datenbank hinterlegt wurde. Sobald jedes Objekt eine Nummer hatte, kam es zu einer Exception während des Schreibvorgangs in der Datenbank, da jedes „Child“ des selben „Parents“ eine identische Nummer erhielt.

Die Lösung war eine statische Map, welche nach der Datenbankabfrage ausgewertet wurde und im Anschluss die neue „Child“-Nummer aufnahm. Die Map ist nur innerhalb eines einzelnen Hochladevorgangs relevant, um die neuen „Child“-Nummern zu cachen und Mehrfachvergaben zu verhindern:

```
//Identifying the highest child number known to the recentChildrenMap
//Required for simultaneous uploads of multiple children to the same family
//Otherwise the system will try to assign the same child number to every object
//Only relevant within one single upload
if (recentChildrenMap.containsKey(familyNumber)){
    int highestRecentChild = 0;
    if (recentChildrenMap.get(familyNumber)[0] instanceof Integer){
        highestRecentChild = (Integer)recentChildrenMap.get(familyNumber)[0];
    }
    if (highestRecentChild >= familyChildNumber){
        highestRecentChild += 1;
        familyChildNumber = highestRecentChild;
    }
}
//Updating the recentChildrenMap, the key is the 7-digit family number, the object array contains the highest known child
number and a timestamp
Object[] currentChildInformation = new Object[2];
currentChildInformation[0] = (Integer)familyChildNumber;
currentChildInformation[1] = new Timestamp(System.currentTimeMillis());
recentChildrenMap.put(familyNumber, currentChildInformation);
```

Ein regelmäßiger flush der Map anhand der Timestamps war ursprünglich geplant, ließ sich auf Grund des Zeitdrucks jedoch nicht umsetzen.

8.3 Testfazit

Der Code ist in einem akzeptablen Zustand für den Einsatz auf dem internen Testsystem des Kunden. Nach mehrstündigen Tests konnten keine weiteren Bugs durch den finalen Logiktest ausfindig gemacht werden.

9. Projektabschluss

Das Projekt ließ sich innerhalb der vorgegebenen Zeit abschließen, jedoch kam es durch den Zeitdruck zu einer suboptimalen Programmierungsarbeit.

Wie in der Entwurfsphase 6.3.2 leicht zu erkennen ist, benutzen beide erstellten Klassen eine fast identische Logik. Auf Grund des Zeitdrucks wurde jedoch lediglich ein großer Teil Code der ersten Klasse (ASEPMDocumentNamingDelegate) kopiert und mit minimalen Anpassungen einiger Variablen für die zweite Klasse (ASPartNumberingRule) verwendet. Wäre hier mehr Zeit gewesen, hätte man große Stücke Code in von beiden Klassen nutzbare Methoden auslagern können.

Ebenfalls geplant war ein regelmäßiger flush der statischen Map (recentChildrenMap) in der Klasse ASEPMDocumentNamingDelegate anhand der gespeicherten Timestamps, doch auch dies ließ sich innerhalb der gegebenen Zeit nicht umsetzen. Tragisch ist es jedoch nicht, da die Liste nur langsam wächst und bei jedem Neustart des Servers natürlich geflusht wird.

Das Endergebnis ist zufriedenstellend, die erwünschte Funktionalität wurde bereitgestellt.

10. Glossar

Access Control	Ver-/Gewährt Zugriff auf im System vorhandene Daten
Apache	Open-source Software (Web Server)
AR	Augmented Reality, überlagern der echten Welt mit digitalen Elementen
Attribut	In Windchill-Kontext: Träger eines Wertes auf einem Objekt
Bibliothek	In Windchill-Kontext: beinhaltet Information, welche nicht fest zu einem Produkt zuzuordnen sind, z.B. Normteile
BOM	Bill of material (Stückliste), Liste aller relevanten Komponenten einer Baugruppe
CAD-Datei	Baugruppe, Komponente oder Zeichnung aus einem CAD-Programm
CAD Data Management	CAD-Daten-Verwaltung, definiert Verhalten für den Umgang mit CAD-Daten
Change Management	Änderungsverwaltung, wird im Geschäftsbereich genutzt um Änderungen ordnungsgemäß umzusetzen und zu dokumentieren
Checkin	In Windchill-Kontext: ereignet sich bei Erstanlage eines Objektes und nach einem Checkout
Checkout	In Windchill-Kontext: benötigt um bestehende Objekte zu verändern
Child	In Projekt-Kontext: CAD-Datei (Komponente), welche auf einem anderen Modell basiert
Creo	Ähnlich Adobe (jedoch für Fertigungsunternehmen), stellt eine Vielzahl verschiedener untereinander nahtlos kompatibler Anwendungen bereit z.B. Creo Parametric (CAD-Programm).
Datenbank-Sequenz	Nach bestimmtem Schema fortlaufende Nummer in Datenbank
EPMDocument	In Windchill-Kontext: Superklasse aller CAD-Dokumente

Hyper-V	Virtualisierungssoftware von Microsoft
IoT	Internet of things (Internet der Dinge), Vernetzung eigenständiger Geräte zu verschiedenen Zwecken (Echtzeit-Fernüberwachung, Datenanalyse etc.)
LDAP	Lightweight Directory Access Protocol, Industriestandard für den Zugriff auf und die Aufbewahrung von Verzeichnisdiensten
MSSQL	Microsoft Sequel Server, lizenzierte Datenbank-Anwendung
OIR	In Windchill-Kontext: Object Initialization Rule, beeinflusst Verhalten von Objekten bei ihrer Erstellung
OOTB	Out-of-the-box, Konfigurations-/Einstellungsmöglichkeiten nach Installation der Software (z.B. Lösung für Problem X ist OOTB nicht verfügbar, eine Java-Customization muss geschrieben werden)
OpenDJ	Open-source Software (LDAP)
Owner-Link	In Windchill-Kontext: engster Beziehungstyp zwischen einem WTPart und einem CAD-Dokument (Komponente oder Baugruppe), ausserdem bei weitem meistgenutzter Beziehungstyp
Parent	In Projekt-Kontext: CAD-Datei (Komponente), welche als Basis für andere Modelle dient
PDM	Product Data Management (Produkt-Daten-Verwaltung), zentrale Verwaltung und Bearbeitung der Produktdaten in einem System
PLM	Product Lifecycle Management (Produkt-Lebenszyklus-Verwaltung), zentrale Verwaltung des gesamten Lebenszyklus eines Produkts (von Erstentwurf über die freigegebene Version bis hin zur obsoleten Version)
Preference	In Windchill-Kontext: zur Laufzeit veränderbare Einstellungen, welche das Verhalten der Anwendung beeinflussen, Anwendung auf verschiedenen Ebenen möglich (Benutzer-, Organisations- oder Applikations-Ebene)
Produkt	In Windchill-Kontext: beinhaltet Informationen, welche fest einer einzelnen Verwendung zugewiesen werden können (nicht-Normteile)
Promotion Process	Freigabeprozess, benötigt um sich in Entwicklung befindliche Objekte ordnungsgemäß in einen freigegebenen Zustand zu bringen
PSI	PTC Solution Installer, Anwendung welche die Installation von PTC-Software leitet
PTC	Software-Hersteller für die Fertigungsindustrie (hauptsächlich)
Versioning	Versionierung, ermöglicht sauberes, gänzlich dokumentiertes Arbeiten mehrerer Leute auf einem Objekt oder Datensatz
VM	Virtuelle Maschine, ein in einer Virtualisierungssoftware laufendes Betriebssystem
VR	Virtual Reality (virtuelle Realität), simulierte Welt, in der Regel durch eine VR-Brille erlebt
Web-basierte Anwendung	Über einen Browser nutzbare Anwendung, nutzt oft Hardware-Ressourcen eines Servers aus der Ferne
Windchill PDMLink	Basisprodukt des Windchill-Portfolios
WTPart	Objekttyp in Windchill, geht meist eine Beziehung mit einer CAD-Datei ein (Baugruppe oder Komponente), indirekt auch mit Zeichnungen. Ein WTPart ist eine Repräsentation eines Gegenstands als Ganzes

11. Quellenverzeichnis

Die Recherche für den Entwurf des Projekts wurde in der PTC Knowledge Base gemacht. Hier findet sich jegliche Dokumentation über die Software, von einfachen Konfigurationsoptionen bis hin zu Lösungsansätzen für häufig auftretende Probleme und Beschreibungen von Bugs. Die PTC Knowledge Base ist Kunden und Partnern mit aktiven Verträgen vorenthalten.

Anwender ohne PTC-Wartungsvertrag können Artikeltitel und -beschreibungen zwar einsehen, jegliche bereitgestellte Informationen sind jedoch nicht einsehbar.

<https://www.ptc.com/en/support/>

How to use autonumbering when creating and uploading a CAD Document in Creo Parametric:

<https://www.ptc.com/en/support/article/CS172880?language=en&posno=4&q=epmdocumentnamingdelegate&source=search>

How to adjust OIRs (OIR Knowledge Hub):

<https://www.ptc.com/en/support/article/CS172537?language=en&posno=1&q=oir%20set%20number&source=search>

How to use a new database sequence:

<https://www.ptc.com/en/support/article/CS38320?language=en&posno=8&q=oir%20set%20number&source=search>

How to create a new Sequence in SQL Server and use it in OIR in Windchill PDMLink:

<https://www.ptc.com/en/support/article/CS42301?source=search>

Is it possible to set the CAD Document number to be the same as the Part number when creating a WTPart in Windchill PDMLink:

<https://www.ptc.com/en/support/article/CS150469?language=en&posno=2&q=Set%20CAD%20Document%20Number%20Same%20As%20Part%20Number&source=search>

12. Anhang

12.1 DB-Sequenz-Template

```
CREATE TABLE wt_sequence_ASPartSeq (dummy CHAR(1),value BIGINT IDENTITY(1000000, 1))
go
CREATE PROCEDURE wt_get_next_sequence_ASPartSeq @returnValue BIGINT OUTPUT
AS
INSERT wt_sequence_ASPartSeq (dummy) VALUES ('x')
SELECT @returnValue = SCOPE_IDENTITY()
Go
```

12.2 OIR WTPart

```
<AttributeValues objType="com.as.AS_Part">
  <!-- set the lifecycle -->
  <AttrValue id="lifeCycle.id" algorithm="com.ptc.core.foundation.lifecycle.server.impl.LifeCycleTemplateAttributeAlgorithm">
    <Arg>AS Part Life Cycle</Arg>
  </AttrValue>
  <!-- set the number to a generated number -->
  <AttrValue id="number" algorithm="ext.as.part.ASPartNumberingRule">
    <Arg>{GEN:wt.enterprise.SequenceGenerator:ASPartSeq:7:0}</Arg>
  <Attr id="IBA|com.as.FAMILY_NUMBER"/>
  <Attr id="IBA|com.as.FAMILY_PARENT"/>
  </AttrValue>
  <!-- specify AttrConstraint tag -->
  <AttrConstraint id="number" algorithm="com.ptc.core.rule.server.impl.GatherAttributeConstraints">
    <Value algorithm="com.ptc.core.rule.server.impl.GetServerAssignedConstraint"/>
  </AttrConstraint>
</AttributeValues>
```

12.3 ASEPMDocumentNamingDelegate

```
package ext.as.cad;

import com.ptc.core.meta.common.IdentifierFactory;
import com.ptc.core.meta.common.TypeIdentifier;
import com.ptc.windchill.uwgm.proesrv.c11n.DocIdentifier;
import com.ptc.windchill.uwgm.proesrv.c11n.EPMDocumentNamingDelegate;
import ext.arsandis.util.LogUtil;
import java.lang.invoke.MethodHandles;
import java.sql.Timestamp;
import java.util.HashMap;
import java.util.Map;
import java.util.regex.Pattern;
import wt.epm.EPMDocument;
import wt.fc.PersistenceHelper;
import wt.fc.QueryResult;
import wt.part.WTPart;
import wt.query.QueryException;
import wt.query.QuerySpec;
import wt.query.SearchCondition;
import wt.services.applicationcontext.implementation.DefaultServiceProvider;
import wt.type.TypedUtilityServiceHelper;
import wt.util.WTException;

public class ASEPMDocumentNamingDelegate implements EPMDocumentNamingDelegate {

    private static final org.apache.log4j.Logger LOGGER = LogUtil.getLoggerFromWTProperties( MethodHandles.lookup().lookupClass() );
    //Name of the parameter responsible for initiating the auto-numbering during the first upload
    private static final String AS_AUTO_NUMBER_PARAMETER = "AS_AUTO_NUMBERING";
    //Required to remember child numbers already pulled during the same upload
    private static Map <String,Object[]> recentChildrenMap = new HashMap<>();

    /**
     * This method is called whenever a CAD file is checked in via Creo Parametric.
     * It can set the common name as well as the number.
     * As the OIRs do not influence new EPM documents checked in via Creo, this is necessary to implement a custom numbering scheme.
     * Once a CAD file is uploaded, all designated parameters are checked and based on the result a number is generated.
     * Files not containing the AS_AUTO_NUMBERING parameter will receive their number via their own filename.
     * Files containing the FAMILY_PARENT value of true will pull a new number and append a "-XX" suffix.
     * Files containing a valid FAMILY_NUMBER value will not pull anew number, rather reuse the specified one and append an incrementing
     suffix.
     * Files without a valid FAMILY_NUMBER or FAMILY_PARENT value will simply pull a new number.
     * @param docIdentifier is the EPM Document pre-upload
     */
    @Override
    public void validateDocumentIdentifier (DocIdentifier docIdentifier) {
        LOGGER.info("Upload event of CAD File: " + docIdentifier.getDocName() + " started!");
        HashMap parameterMap = docIdentifier.getParameters();

        //Checking for the Assessment Services parameter to skip any Sensus CAD files
        if (parameterMap.containsKey(AS_AUTO_NUMBER_PARAMETER)){
            //Checking the value to skip any CAD files that are not supposed to be autonumbered
            if ((Boolean)parameterMap.get(AS_AUTO_NUMBER_PARAMETER)){
                //Shortening the name of the EPM Document to at most 60 characters should it be longer
                String docName = docIdentifier.getDocName();
                if (docName.length() > 60) {
                    LOGGER.error("Document name is above 60 character limit!");
                    docIdentifier.setDocName(docIdentifier.getDocName().substring(0, 60));
                }
                //Setting Regexes, parameter names, type identifier strings and other variables required for the numbering process
                final String FAMILY_NUMBER_REGEX = "\\d{7}";
                final String FAMILY_CHILD_REGEX = "\\d{7}-\\d{2,3}";
                final String AS_PART_IDENTIFIER = "WCTYPE|wt.part.WTPart|com.as.AS_Part";
                final String AS_CAD_IDENTIFIER =
                    "WCTYPE|wt.epm.EPMDocument|com.sensus.DefaultEPMDocument|com.as.AS_CADDocument";
                final String AS_CAD_SEQUENCE = "ASPartSeq";
                final String FAMILY_PARENT_PARAMETER = "FAMILY_PARENT";
                final String FAMILY_NUMBER_PARAMETER = "FAMILY_NUMBER";
                final String PARENT_SUFFIX="-XX";
                String numberSuffix="";
            }
        }
    }
}
```



```

String epmNumber="";
String familyNumber = "";
boolean familyParent = false;

if (parameterMap.containsKey(FAMILY_NUMBER_PARAMETER)){
    familyNumber = (String)parameterMap.get(FAMILY_NUMBER_PARAMETER);
}
if (parameterMap.containsKey(FAMILY_PARENT_PARAMETER)){
    familyParent = (boolean)parameterMap.get(FAMILY_PARENT_PARAMETER);
}
//Checking whether to set this as a family parent number or not
if (familyNumber.isEmpty() && familyParent){
    numberSuffix = PARENT_SUFFIX;
//Checking whether to set this as a family child number or not
} else if (Pattern.matches(FAMILY_NUMBER_REGEX, familyNumber) && !familyParent){
    //Checking whether a family parent exists as an EPM Document
    try {
        String familyParentNumber = familyNumber + PARENT_SUFFIX;
        //Required to specify a subtype in a searchcondition
        final IdentifierFactory IDENTIFIER_FACTORY = (IdentifierFactory) DefaultServiceProvider.getService(IdentifierFactory.class,
"logical");
        //Generating AS_CADDocument typeidentifier
        TypedIdentifier tidEpm = (TypedIdentifier) IDENTIFIER_FACTORY.get(AS_CAD_IDENTIFIER);
        //Generating searchcondition specific to AS_CADDocument
        SearchCondition AsCadSC = TypedUtilityServiceHelper.service.getSearchCondition(tidEpm, true);
        QuerySpec qsParentEpm = new QuerySpec( EPMDocument.class );
        int idxParentEpm = qsParentEpm.addClassList(EPMDocument.class, true);
        qsParentEpm.appendWhere(AsCadSC, new int[] { idxParentEpm });
        qsParentEpm.appendAnd();
        qsParentEpm.appendWhere(new SearchCondition(EPMDocument.class, EPMDocument.NUMBER, SearchCondition.EQUAL,
familyParentNumber ), new int[] { idxParentEpm });
        QueryResult qrParentEpm = PersistenceHelper.manager.find(qsParentEpm);
        boolean parentExists = false;

        //Checking whether a parent EPM Document was found or not
        if (qrParentEpm.size()>=1){
            parentExists = true;
        //In case no parent EPM Document was found, a query for a parent WTPart will be run
        } else {
            TypedIdentifier tidPart = (TypedIdentifier) IDENTIFIER_FACTORY.get(AS_PART_IDENTIFIER);
            SearchCondition AsPartSC = TypedUtilityServiceHelper.service.getSearchCondition(tidPart, true);
            QuerySpec qsParentPart = new QuerySpec( WTPart.class );
            int idxParentPart = qsParentPart.addClassList( WTPart.class, true);
            qsParentPart.appendWhere(AsPartSC, new int[] { idxParentPart });
            qsParentPart.appendAnd();
            qsParentPart.appendWhere(new SearchCondition(WTPart.class, WTPart.NUMBER, SearchCondition.EQUAL,
familyParentNumber ), new int[] { idxParentPart });
            QueryResult qrParentPart = PersistenceHelper.manager.find(qsParentPart);

            if (qrParentPart.size()>=1){
                parentExists = true;
            }
        }
    }
    //Identification of highest child number, should a parent WTPart or EPM Document exist
    //In case no parent is found, a new number will be pulled and the EPM Document will not receive a family child number
    if (parentExists){
        //Running a query for all children in the database
        QuerySpec qsChildren = new QuerySpec( EPMDocument.class );
        int idxChildren = qsChildren.addClassList(EPMDocument.class, true);
        qsChildren.appendWhere(AsCadSC, new int[] { idxChildren });
        qsChildren.appendAnd();
        qsChildren.appendWhere(new SearchCondition(EPMDocument.class, EPMDocument.NUMBER, SearchCondition.LIKE,
familyNumber+"%", new int[] { idxChildren }); //% appended
        QueryResult qrChildren = PersistenceHelper.manager.find(qsChildren);
        //Identifying the highest child number known to the database
        int familyChildNumber = 1;
        while (qrChildren.hasMoreElements()){
            String currentEPMNumber = ((EPMDocument)qrChildren.nextElement()).getNumber();
            if (Pattern.matches(FAMILY_CHILD_REGEX, currentEPMNumber)){
                int currentChildNumber = Integer.parseInt(currentEPMNumber.substring(8));
                if (currentChildNumber >= familyChildNumber){

```

```

        familyChildNumber = currentChildNumber + 1;
    }
}

//Identifying the highest child number known to the recentChildrenMap
//Required for simultaneous uploads of multiple children to the same family
//Otherwise the system will try to assign the same child number to every object
//Only relevant within one single upload
if (recentChildrenMap.containsKey(familyNumber)){
    int highestRecentChild = 0;
    if (recentChildrenMap.get(familyNumber)[0] instanceof Integer){
        highestRecentChild = (Integer)recentChildrenMap.get(familyNumber)[0];
    }
    if (highestRecentChild >= familyChildNumber){
        highestRecentChild += 1;
        familyChildNumber = highestRecentChild;
    }
}

//Updating the recentChildrenMap, the key is the 7-digit family number, the object array contains the highest known child
number and a timestamp
Object[] currentChildInformation = new Object[2];
currentChildInformation[0] = (Integer)familyChildNumber;
currentChildInformation[1] = new Timestamp(System.currentTimeMillis());
recentChildrenMap.put(familyNumber, currentChildInformation);
//Adding required characters to the suffix
if (familyChildNumber < 10){
    numberSuffix = "-0" + Integer.toString(familyChildNumber);
} else {
    numberSuffix = "-" + Integer.toString(familyChildNumber);
}
epmNumber = familyNumber;
}
} catch (QueryException ex) {
    LOGGER.error("Failed to create QuerySpec for identifying Family Membership for CAD Document: " +
docIdentifier.getDocName(), ex);
} catch (WTEException ex) {
    LOGGER.error("Failed to generate QueryResult for identifying Family Membership for CAD Document: " +
docIdentifier.getDocName(), ex);
}
}
}

//Pull new number if necessary
if (epmNumber.isEmpty()){
    try {
        epmNumber = wt.fc.PersistenceHelper.manager.getNextSequence(AS_CAD_SEQUENCE);
    } catch (WTEException ex) {
        LOGGER.error("Could not generate number from sequence: " + AS_CAD_SEQUENCE, ex);
    }
}

//Append suffix to the number and set the number of the EPM Document
epmNumber += numberSuffix;
docIdentifier.setDocNumber(epmNumber);
}
}
}
}

```

12.4 ASPartNumberingRule

```
package ext.as.part;
```

```
import com.ptc.core.meta.common.IdentifierFactory;
import com.ptc.core.meta.common.TypeIdentifier;
import ext.arsandis.util.LogUtil;
import java.lang.invoke.MethodHandles;
import java.util.regex.Pattern;
import wt.epm.EPMDocument;
import wt.part.WTPart;
import wt.fc.PersistenceHelper;
import wt.fc.QueryResult;
import wt.inf.container.WTContainerRef;
import wt.query.QueryException;
import wt.query.QuerySpec;
```

```

import wt.query.SearchCondition;
import wt.rule.algorithm.RuleAlgorithm;
import wt.services.applicationcontext.implementation.DefaultServiceProvider;
import wt.type.TypedUtilityServiceHelper;
import wt.util.WTException;

/**
 * This method is called whenever a WTPart is created in the Assessment Services Windchill Context.
 * It is a custom RuleAlgorithm and is called in the OIR during the setting of the number.
 * This code will also be called via the custom folder-toolbar-action "Generate Numbers" when renumbering WTParts.
 * 2 IBA values from the WTPart are passed over.
 * Files containing the FAMILY_PARENT value of true will pull a new number and append a "-XX" suffix.
 * Files containing a valid FAMILY_NUMBER value will not pull a new number, rather reuse the specified one and append an incrementing
 * suffix.
 * Files without a valid FAMILY_NUMBER or FAMILY_PARENT value will simply pull a new number.
 */
public class ASPartNumberingRule implements RuleAlgorithm {

    private static final org.apache.log4j.Logger LOGGER = LogUtil.getLoggerFromWTPProperties(MethodHandles.lookup().lookupClass());

    @Override
    public Object calculate(Object args[], WTContainerRef container) throws WTException {

        //Setting regular expressions, parameter names, type identifier strings and other variables required for the numbering process
        final String FAMILY_NUMBER_REGEX = "\\d{7}";
        final String FAMILY_CHILD_REGEX = "\\d{7}-\\d{2,3}";
        final String AS_PART_IDENTIFIER = "WCTYPE|wt.part.WTPart|com.as.AS_Part";
        final String AS_CAD_IDENTIFIER = "WCTYPE|wt.epm.EPMDocument|com.sensus.DefaultEPMDocument|com.as.AS_CADDocument";
        final String AS_PART_SEQUENCE = "ASPartSeq";
        final String PARENT_SUFFIX = "-XX";
        String numberSuffix = "";
        String partNumber = "";
        String familyNumber = "";
        boolean familyParent = false;

        if (args[1] instanceof String && args[1] != null) {
            familyNumber = (String) args[1];
        }
        if (args[2] instanceof Boolean) {
            familyParent = (boolean) args[2];
        }

        //Checking whether to set this as a family parent number or not
        if (familyNumber.isEmpty() && familyParent) {
            numberSuffix = PARENT_SUFFIX;
            //Checking whether to set this as a family child number or not
        } else if (Pattern.matches(FAMILY_NUMBER_REGEX, familyNumber) && !familyParent) {
            //Checking whether a family parent exists as a WTPart
            try {
                String familyParentNumber = familyNumber + PARENT_SUFFIX;
                //Required to specify a subtype in a searchcondition
                final IdentifierFactory IDENTIFIER_FACTORY = (IdentifierFactory) DefaultServiceProvider.getService(IdentifierFactory.class,
"logical");
                //Generating AS_Part typeidentifier
                TypedIdentifier tidPart = (TypedIdentifier) IDENTIFIER_FACTORY.get(AS_PART_IDENTIFIER);
                //Generating searchcondition specific to AS_Part
                SearchCondition AsPartSC = TypedUtilityServiceHelper.service.getSearchCondition(tidPart, true);
                QuerySpec qsParentPart = new QuerySpec(WTPart.class);
                int idxParentPart = qsParentPart.addClassList(WTPart.class, true);
                qsParentPart.appendWhere(AsPartSC, new int[]{idxParentPart});
                qsParentPart.appendAnd();
                qsParentPart.appendWhere(new SearchCondition(WTPart.class, WTPart.NUMBER, SearchCondition.EQUAL,
familyParentNumber), new int[]{idxParentPart});
                QueryResult qrParentPart = PersistenceHelper.manager.find(qsParentPart);
                boolean parentExists = false;

                //Checking whether a parent WTPart was found or not
                if (qrParentPart.size() >= 1) {
                    parentExists = true;
                    //In case no parent WTPart was found, a query for a parent EPMDocument will be run
                } else {
                    TypedIdentifier tidEpm = (TypedIdentifier) IDENTIFIER_FACTORY.get(AS_CAD_IDENTIFIER);

```

```

SearchCondition AsCadSC = TypedUtilityServiceHelper.service.getSearchCondition(tidEpm, true);
QuerySpec qsParentEpm = new QuerySpec(EPMDocument.class);
int idxParentEpm = qsParentEpm.addClassList(EPMDocument.class, true);
qsParentEpm.appendWhere(AsCadSC, new int[]{idxParentEpm});
qsParentEpm.appendAnd();
qsParentEpm.appendWhere(new SearchCondition(EPMDocument.class, EPMDocument.NUMBER, SearchCondition.EQUAL,
familyParentNumber), new int[]{idxParentEpm});
QueryResult qrParentEpm = PersistenceHelper.manager.find(qsParentEpm);

if (qrParentEpm.size() >= 1) {
    parentExists = true;
}
}
//Identification of highest child number, should a parent WTPart or EPM Document exist
//In case no parent is found, a new number will be pulled and the EPM Document will not receive a family child number
if (parentExists) {
    //Running a query for all children in the database
    QuerySpec qsChildren = new QuerySpec(WTPart.class);
    int idxChildren = qsChildren.addClassList(WTPart.class, true);
    qsChildren.appendWhere(AsPartSC, new int[]{idxChildren});
    qsChildren.appendAnd();
    qsChildren.appendWhere(new SearchCondition(WTPart.class, WTPart.NUMBER, SearchCondition.LIKE, familyNumber + "%"),
new int[]{idxChildren}); //or * appended to familyMember to be tested
    QueryResult qrChildren = PersistenceHelper.manager.find(qsChildren);
    //Identifying the highest child number known to the database
    int familyChildNumber = 1;
    while (qrChildren.hasMoreElements()) {
        String currentPartNumber = ((WTPart) qrChildren.nextElement()).getNumber();
        if (Pattern.matches(FAMILY_CHILD_REGEX, currentPartNumber)) {
            int currentChildNumber = Integer.parseInt(currentPartNumber.substring(8));
            if (currentChildNumber >= familyChildNumber) {
                familyChildNumber = currentChildNumber + 1;
            }
        }
    }
    //Adding required child number characters to the suffix
    if (familyChildNumber < 10) {
        numberSuffix = "-0" + Integer.toString(familyChildNumber);
    } else {
        numberSuffix = "-" + Integer.toString(familyChildNumber);
    }
    partNumber = familyNumber;
}
} catch (QueryException ex) {
    LOGGER.error("Failed to create QuerySpec for identifying Family Membership for new WTPart", ex);
} catch (WTEException ex) {
    LOGGER.error("Failed to generate QueryResult for identifying Family Membership for new WTPart", ex);
}
}
//Pull new number if necessary
if (partNumber.isEmpty()) {
    try {
        partNumber = wt.fc.PersistenceHelper.manager.getNextSequence(AS_PART_SEQUENCE);
    } catch (WTEException ex) {
        LOGGER.error("Could not generate number from sequence: " + AS_PART_SEQUENCE, ex);
    }
}
//Append suffix to the number and return the number for the WTPart
partNumber += numberSuffix;
return partNumber;
}
}

```

12.5 Nummerierungstestplan

Prerequisites

- Open Creo Parametric

- File > Manage Session > Server Management > Connect to Server
Name: PLM-DEV | Address: <https://plm-dev.xylem.com/Windchill>
- Log in as asmechanical2
- Set as primary server
- Go to File > Options > Configuration Editor
- change value of "template_solidpart" to Libraries/AS Test Library/asdemostartpart.prt
- change value of "template_drawing" to Libraries/AS Test Library/asdemostartdrawing.drw
- change value of "template_designasm" to Libraries/AS Test Library/asdemostartassembly.asm

Create CAD file in Creo. Check in to Windchill without WTPart. Associate WTParts later. Create WTPart during associating.

- Log into Creo as asmechanical2
- Create a new CAD file
File name: ClickScriptScenario1-1Cad
- Auto Check In

Variation: Do the same process with an Assembly

- Repeat the above process, but instead of using/creating a CAD file, create an Assembly in Creo

Variation: Do the same process with a Drawing

- Repeat the above process, but instead of using/creating a CAD file, create an Assembly in Creo

Create CAD file in Creo. Check in to Windchill without WTPart. Associate WTParts later. Create WTPart before associating.

- Log into Creo as asmechanical2
- Create a new CAD file
File name: ClickScriptScenario1-2Cad
- Check In
- In Windchill commonspace create new AS Part
- Right Click cad file -> Edit Associations

Create CAD file in Creo. Check in to Windchill without WTPart. Auto Associate WTParts.

- Log into Creo as asmechanical2
- Create a new CAD file
File name: ClickScriptScenario1-3Cad
- Check In
- In Workspace Right Click cad file -> Auto Associate (creates new part automatically for CAD file, only works in workspace)
- check in the newly created part
- check CAD file associations in view information page

Create CAD file in Creo. Check in to Windchill with Auto Associate WTParts

- Log into Creo as asmechanical2

- Create a new CAD file
File name: ClickScriptScenario1-4Cad
- Custom Check In
- Under “Set Options” check “Auto Associate Parts”

Create WTPart first. Create CAD later and associate corresponding parts.

- In Windchill create WTPart named clickscriptscenario1-5
- In Creo create new CAD file named clickscriptscenario1-5
- check in CAD file
- In Windchill right click CAD file > Edit Associations > Associate CAD file to the WTPart with an owner link
- check in CAD file

Create WTPart and CAD at the same time.

- In Windchill create WTPart named clickscriptscenario1-6
- check box for ‘Create CAD Document’
- select Creo as Authoring Application
- select Template “asdemocadtemplate.prt”

Variation: Number Attribute: Same as Part Number

- under Attributes select Number: Same as Part Number
- click finish

Variation: Number Attribute: Not Same as Part Number

- under Attributes select Number: 500

Variation: Create Assembly with Same Number

- select Type “AS CAD Document”
- select Template “asdemoassemblytemplate.asm”
- select Number: Same as Part Number

Variation: Create Assembly with manual Number

- select Type “AS CAD Document”
- select Template “asdemoassemblytemplate.asm”
- select Number: 004

Autonumbering with Families

- In Creo Parametric log in as user: asmechanical2
- create new cad file “clickscriptscenario1-8-1”
- go to tools > Parameters and at “FAMILY_PARENT” select YES
- custom check in > Auto Associate Parts to CAD docs
- create a new cad file “clickscriptscenario1-8-2”
- go to tools > Parameters and at “FAMILY_NUMBER” enter the Parent part number (in my case 1000199) (don’t enter the suffix “-XX”)
- custom check in > Auto Associate Parts to CAD docs

- repeat last 3 steps. cad file name “clickscriptscenario1-8-3” Create CAD docs without Auto Associating Parts in Creo. Associate parts later. Create parts in “Edit Associations” wizard.
- repeat the main process without Auto Associating Parts within Creo. cad file names: clickscriptscenario1-8-5; clickscriptscenario1-8-6
- Edit Associations of both parts. In the Edit Associations wizard use “Create new part for Association”

Create CAD docs without Auto Associating Parts in Creo. Create Parts later, then Associate.

- repeat the main process without Auto Associating Parts within Creo. cad file names: clickscriptscenario1-8-7; clickscriptscenario1-8-8
- Create WTParts
- Edit Associations of both CAD files. In the Edit Associations wizard use “Existing Part”

Autonumbering with Families and Family Tables feature of Creo Parametric

- Create CAD file inside Creo Parametric
- change FAMILY_PARENT parameter to yes
- check in
- check out
- create family table and add the two parameters FAMILY_PARENT and FAMILY_NUMBER as columns
- for every **instance** change FAMILY_PARENT to NO and add the FAMILY_NUMBER
- verify instances
- check in

Variance 1: Create CAD file and family table at the same time

- that doesn’t work, because for the family numbering to work the family parent already needs to be uploaded to windchill (so that the children can scan for its number). As you create all parts at the same time in a family table there is no family parent to scan its number from

Variance 2: Create CAD file, set it as family parent, check it into Windchill, check it out and create family table

now we try to trick the system by

- creating a cad part within creo parametric
- set the FAMILY_PARENT attribute of the part to yes
- check in CAD file
- check out CAD file
- create family table
- the parent part needs to have the FAMILY_PARENT attribute set to YES and the children need to have the part number set, that we got by checking it into windchill.