Prüfungsteil A

Prüfling (private Anschrift):	Ausbildungsbetrieb:

Bestätigung über durchgeführte Projektarbeit

diese Bestätigung ist mit der Projektdokumentation einzureichen

Ausbildungsberuf (bitte unbedingt angeben):

Projektbezeichnung:

Projektbeginn:_____Projektfertigstellung:_____Zeitaufwand in Std.:_____

Bestätigung der Ausbildungsfirma:

Wir bestätigen, dass der/die Auszubildende das oben bezeichnete Projekt einschließlich der Dokumentation im Zeitraum

vom:_____ bis: _____ selbständig ausgeführt hat.

Projektverantwortliche(r) in der Firma:

_____

Vorname Name Telefon Unterschrift

Ausbildungsverantwortliche(r) in der Firma:

_____

Vorname Name Telefon Unterschrift

Eidesstattliche Erklärung:

Ich versichere, dass ich das Projekt und die dazugehörige Dokumentation selbständig erstellt habe.

Ort und Datum: _____ Unterschrift des Prüflings:_____

Abschlussprüfung Winter 2018

Fachinformatiker für Anwendungsentwicklung

Dokumentation zur betrieblichen Projektarbeit

Order Comment Tool

Tool zu Unterstützung der Produktionsplaner

Abgabetermin: Mannheim, den 15.12.2018

Prüfungsbewerber:
Thomas Pöhlmann
Schwingstraße 10
68199 Mannheim

Ausbildungsbetrieb:
Camelot ITLab GmbH
Theodor-Heuss-Anlage. 12
68165 Mannheim

Inhaltsverzeichnis

Abbildungsverzeichnis **III**

Tabellenverzeichnis **IV**

Abkürzungsverzeichnis **V**

Transaktionsverzeichnis **VI**

1 Einleitung **1**
1.1 Vorstellung des Betriebs und meiner Selbst . . . 1
1.2 Projektziel . . . 1
1.3 Projektumfeld . . . 1
1.4 Projektbeteiligte Personen . . . 2
1.5 Projektbegründung . . . 2

2 Projektplanung **3**
2.1 Projektphasen . . . 3
2.2 Ressourcenplanung . . . 3
2.3 Entwicklungsprozess . . . 3

3 Analysephase **4**
3.1 Ist-Analyse . . . 4
3.2 Wirtschaftlichkeitsanalyse . . . 4
3.2.1 Projektkosten . . . 4
3.2.2 Amortisationsdauer . . . 5
3.3 Nutzwertanalyse . . . 7
3.4 Anwendungsfälle . . . 7
3.5 Qualitätsanforderungen . . . 7
3.6 Lastenheft/Fachkonzept . . . 8
3.7 Zwischenstand . . . 8

4 Entwurfsphase **9**
4.1 Zielplattform . . . 9
4.2 Architekturdesign . . . 9
4.3 Entwurf des Userinterface . . . 9
4.4 Datenmodell . . . 10
4.5 Maßnahmen zur Qualitätssicherung . . . 11
4.6 Pflichtenheft/Datenverarbeitungskonzept . . . 11
4.7 Zwischenstand . . . 11

5 Implementierungsphase **12**

5.1 Iterationsplanung . . . 12
5.2 Implementierung der Datenstruktur ECC . . . 12
5.3 Implementierung der Benutzeroberfläche ECC . . . 12
5.4 Implementierung PBO und PAI . . . 12
5.5 Implementierung der Geschäftslogik ECC . . . 13
5.6 Implementierung der COR Erweiterung . . . 13
5.7 Implementierung der Datenstruktur APO . . . 14
5.8 Implementierung der Benutzeroberfläche APO . . . 14
5.9 Implementierung der Geschäftslogik APO . . . 14
5.10 Implementierung der RRP3 Erweiterung . . . 15
5.11 Zwischenstand . . . 15

6 Qualitätssicherung 16
6.1 Code Reviews während des Projekts . . . 16
6.2 Manuelle Tests . . . 16
6.3 Bug Fixing . . . 16

7 Fazit 17
7.1 Soll-/Ist-Vergleich . . . 17
7.2 Ausblick . . . 17

Quellenverzeichnis 18

A Anhang i
A.1 Detaillierte Zeitplanung . . . i
A.2 Ressourcen Übersicht . . . ii
A.3 Lastenheft (Auszug) . . . iii
A.4 Iterationsplan . . . iv
A.5 Use-Case-Diagramm . . . v
A.6 Pflichtenheft (Auszug) . . . v
A.7 Dictionary-Objekte . . . vii
A.7.1 Tabellentypen . . . vii
A.7.2 Strukturen . . . vii
A.7.3 Datenelemente . . . viii
A.7.4 Domänen . . . viii
A.8 Screenshots der Anwendung . . . ix
A.9 Programmierrichtlinien . . . xix
A.9.1 Benennung . . . xix
A.9.2 Formatierung . . . xix
A.10 Klassendiagramm . . . xx

Abbildungsverzeichnis

Abbildung 1 Vereinfachtes Datenbank-Model der /CAMELOT/OC_COMT 10
Abbildung 2 Vereinfachtes Datenbank-Modell der /CAMELOT/OC_SET 10
Abbildung 3 Vereinfachtes Datenbank-Modell der /SAPAPO/ORDFLDS 10
Abbildung 4 Use-Case-Diagramm v
Abbildung 5 Ändern des Feldnamens für den Kommentar in der AUFK Datenbanktabelle . ix
Abbildung 6 Selektionsbildschirm für den Maintenance Screen im ECC x
Abbildung 7 Aufträge im ALV auf dem Maintenance Screen im ECC xi
Abbildung 8 Selektions Popup auf dem Maintenance Screen xii
Abbildung 9 Selektionsbildschirm für den Maintenance Screen im APO xiii
Abbildung 10 Aufträge im ALV auf dem Maintenance Screen im APO xiv
Abbildung 11 Kommentare in der RRP3 xv
Abbildung 12 Kommentare in der RRP3 zum Editieren xvi
Abbildung 13 Kommentar in der COR3 xvii
Abbildung 14 Kommentar in der COR2 zum Bearbeiten xviii
Abbildung 15 Einstellungen des Pretty-Printers xix
Abbildung 16 Klassendiagramm xx

Tabellenverzeichnis

Tabelle 1 Beteiligte Personen 2
Tabelle 2 Zeitplanung 3
Tabelle 3 Hardwarekosten 5
Tabelle 4 Projektkosten 5
Tabelle 5 Nutzwertanalyse 7
Tabelle 6 Zwischenstand nach der Analysephase 8
Tabelle 7 Zwischenstand nach der Entwurfsphase 11
Tabelle 8 Zwischenstand nach der Implementierungsphase 15
Tabelle 9 Soll-/Ist-Vergleich der Projektphasen Zeitplanung 17

Abkürzungsverzeichnis

ABAP Advanced Business Application Programming

ALV ABAP List View

APO Advanced Planning and Optimization

BAdI Business Add-In

ERP Enterprise-Resource-Planning

ECC Enterprise-Resource-Planning (ERP) Central Component

GUI Graphical User Interface

MVC Model View Controller

PBO Process Before Output

PAI Process After Input

RFC Remote Function Call

UML Unified Modeling Language

Transaktionsverzeichnis

COR1-3:
Diese Transaktion ist zum Erstellen, Bearbeiten und Anzeigen von Prozessaufträgen.

/SAPAPO/RRP3:
Mithilfe der Produktsicht lassen sich zu einer vorher definierten Produkt-Werkskombination alle Forecasts, Prozessaufträge, Planaufträge, Produktionsaufträge, Kaufanforderungen und vieles mehr anzeigen.

CMOD:
Mittels dieser Transaktion können Erweiterungspunkte ausgewählt und implementiert werden.

se18:
Diese Transaktion dient zur Implementierung von Business Add-In (BAdI)s.

se80
Die se80 Transaktion ist die in meinem Projekt am meisten genutzte Transaktion. Mit ihr wird programmiert und es können Datenbanktabellen und Dictionary-Objekte erstellt werden.

1 Einleitung

Das folgende Projekt ist mein IHK-Abschlussprojekt, welches im Rahmen meiner Ausbildung zum Fachinformatiker im Bereich Anwendungsentwicklung durchgeführt wurde.

1.1 Vorstellung des Betriebs und meiner Selbst

Am 01. März 2017 habe ich meine Ausbildung bei der Spreitzenbarth Consultants GmbH begonnen, einem Unternehmen mit 50 Mitarbeitern und Fokus auf Supply Chain Management. Dort habe ich hauptsächlich Web Anwendungen mit ASP.NET MVC und AngularJs sowie Windows Applikationen mit C# programmiert. Aufgrund der Firmenauflösung musste ich meinen Ausbildungsbetrieb wechseln und bin seit 01. Juni 2018 bei der Camelot ITLab GmbH beschäftigt. Hier beschäftige ich mich mit dem gesamten SAP Umfeld und lerne die Programmiersprache Advanced Business Application Programming (ABAP).

1.2 Projektziel

Im SAP Advanced Planning and Optimization (APO) bzw. im SAP ERP Central Component (ECC) gibt es verschiedene Arten von Aufträgen. Ein Auftrag im allgemeinen umfasst Produkte, Mengen, Ressourcen(Maschinen) und Zeit. Es gibt Planaufträge, eine Art Beschaffungsvorschlag, welche als internes planerisches Element genutzt werden und jederzeit umgebucht werden können. Diese werden dann Richtung Gegenwart zu Prozess- bzw. Produktionsaufträgen konvertiert, welche dann fix gebucht sind und nicht mehr geändert werden können. Der Planer hat die Möglichkeit, Prozess-, Produktions- und Plan-Aufträge manuell anzulegen bzw. vorhandene Planaufträge zu editieren. Allerdings ist es SAP Standard nicht möglich, Notizen oder Beschreibungen für diese zu hinterlegen. Projektziel ist die Erstellung von Programmen, die es dem Planer ermöglichen, Kommentare für Aufträge zu erstellen und anzusehen. Außerdem sollen bereits vorhandene Programme entsprechend erweitert werden, um den Planer bestmöglich zu unterstützen. Da dieses Projekt ein internes Projekt ist, wurden alle Anforderungen in innerbetrieblichen Besprechungen erarbeitet. Bei der Implementierung wurde besonders darauf geachtet, einen leicht zu erweiternden Code zu produzieren, da in zukünftigen Versionen noch weitere Feature hinzugefügt werden sollen, wie z.B. ein Kommentarverlauf, sodass der Planer auch vorherige Versionen von Kommentaren anschauen kann.

1.3 Projektumfeld

Die Camelot ITLab GmbH ist eine im SAP-Umfeld tätige Unternehmensberatung, die sowohl funktionale als auch technische Implementierungen von Geschäftsprozessen umsetzt. Die Abteilung SCM Solution Development innerhalb der Camelot ITLab GmbH, in deren Umfeld auch dieses Projekt umgesetzt wurde, ist für die softwareseitige Implementierung technischer Anforderungen zuständig. Der

Schwerpunkt der Abteilung ist Supply Chain Management, im Speziellen Produktions- und Feinplanung.

1.4 Projektbeteiligte Personen

Tabelle 1 zeigt alle Personen, die an dem Projekt beteiligt waren.

Name	Funktion bzw. Position	Rolle im Projekt
Thomas Pöhlmann	Auszubildender Fachinformatiker für Anwendungsentwicklung	Projektleiter und Auftragnehmer
Florian von der Weth	Senior Consultant	Code Review, Auftraggeber
Florian P.	Junior Consultant	Code Review
Julian P.	Consultant	Code Review, Auftraggeber

Tabelle 1: Beteiligte Personen

1.5 Projektbegründung

Da die im Projektziel beschriebenen Funktionalitäten für den Planer äußerst hilfreich sind und diese schon oft von Kunden angefragt wurden, da es etwas Vergleichbares auf dem Markt bzw. im SAP Standard noch nicht gibt, wurde dieses Projekt von der Camelot ITLab GmbH angenommen.

2 Projektplanung

2.1 Projektphasen

Für das Projekt standen 70 Stunden zur Verfügung. Diese waren sowohl für die Umsetzung des Projekts als auch für die Dokumentation gedacht. Vor Projektbeginn wurde das Projekt in mehrere Phasen gesplittet und die Stunden aufgeteilt. Tabelle 2 zeigt eine grobe Übersicht über die einzelnen Phasen und die geplante Zeit.

Projektphase	Geplante Zeit in Stunden
Anforderungsaufnahme	2
Planung	4
Analysephase	6
Entwurfsphase	6
Implementierungsphase	23
Testphase/Qualitätssicherung	6
Fazit	3
Dokumentationsphase	20
Gesamt	**70**

Tabelle 2: Zeitplanung

Eine detailliertere Zeitplanung findet sich im Anhang A.1: Detaillierte Zeitplanung auf Seite i.

2.2 Ressourcenplanung

In der Ressourcenübersicht Anhang A.2: Ressourcen Übersicht auf Seite ii sind alle Ressourcen aufgelistet, die für das Projekt eingesetzt wurden. Damit sind sowohl Hardware, Software und das Personal gemeint. Sie diente dazu, vorab zu überprüfen, ob alle benötigten Mittel zur Verfügung stehen und um bei den beteiligten Personen entsprechen Zeit für z.B. Code Reviews zu buchen. Es wurde darauf geachtet, dass nur Software zum Einsatz kommt, welche entweder kostenfrei (z.B. als Freeware oder Open Source) angeboten wird oder für die die Camelot ITLab GmbH bereits Lizenzen besitzt. Dadurch sollten die Projektkosten möglichst gering gehalten werden.

2.3 Entwicklungsprozess

Als Entwicklungsprozess wurde ein agiler Entwicklungsprozess gewählt, sodass während der Implementierung nach jeder Iterationsphase eine Rücksprache mit dem Ausbilder, Kunden und der Entwicklungsabteilung erfolgte. Aufgrund dieser Tatsache wurde bei der Projektplanung auch relativ wenig Zeit für die Entwurfsphase veranschlagt, da sich Teile dieser Phase erst während der Entwicklung ergaben. Die stetige Kommunikation mit der Entwicklungsabteilung förderte das Erzielen eines besseren Resultats.

3 Analysephase

3.1 Ist-Analyse

Das Projekt wurde im Umfeld der Produktionsplanung durchgeführt und bezieht sich auf Plan-, Prozess- und Produktionsaufträge. Der Planer hat im SAP APO die Möglichkeit, Zugangs- und Abgangselemente manuell anzulegen. In Folgeprozessen kann es zu Problemen kommen, wenn die Gründe dieser Planänderungen nicht sichtbar sind. Bisher müssen solche Änderungen ohne Systemunterstützung per Email oder auf anderem Weg allen Beteiligten mitgeteilt werden.

3.2 Wirtschaftlichkeitsanalyse

Aufgrund der Probleme, die bereits in der Projektbegründung und der Ist-Analyse beschrieben wurden, ist ersichtlich, wie wichtig und hilfreich dieses Projekt ist. Trotzdem wurde im folgenden Abschnitt analysiert, ob die Umsetzung auch aus wirtschaftlichen Gesichtspunkten gerechtfertigt ist.

3.2.1 Projektkosten

Es wurden neben den Kosten, die für Hardware und Software anfallen, auch die Personalkosten berücksichtigt.
Da die genauen Personalkosten Betriebsgeheimnis sind, wurde die Kalkulationen mit groben Stundensätzen durchgeführt. Der Stundensatz eines Auszubildenden ergab sich aus dem Monatsgehalt von ca. 1000€.

$$365 \text{ Tage/Jahr} \cdot \frac{5}{7} - 30 Tage = 230 \text{ Tage/Jahr} \tag{1}$$

$$8 \text{ h/Tag} \cdot 230 \text{ Tage/Jahr} = 1840 \text{ h/Jahr} \tag{2}$$

$$1000 \,€/\text{Monat} \cdot 12 \text{ Monate/Jahr} = 12000 \,€/\text{Jahr} \tag{3}$$

$$\frac{12000 \,€/\text{Jahr}}{1840 \text{ h/Jahr}} \approx 6{,}52 \,€/\text{h} \tag{4}$$

Der Rechnung zufolge beträgt der Stundensatz eines Azubis ca. 7€. Die Kosten eines Mitarbeiters wurden auf 25€ die Stunde geschätzt.
Um die anfallenden Hardwarekosten zu berechnen, wurden die Anschaffungskosten aller Geräte summiert und dann durch die durchschnittliche Lebenszeit in Arbeitsstunden, in diesem Fall 3 * 365 * 5/7 * 8, geteilt.

Tabelle 3 zeigt die Hardwarekosten, die für die Geräte eines Mitarbeiters anfallen.

Gerät	Anzahl	Anschaffungskosten	Gesamt
Lenovo ThinkPad T450s	1	1200€	1200€
Dell 24 P2419H 61	2	170€	340€
Maus und Tastatur	1	20€	20€
Lenovo Dockingstation	1	120€	120€
Gesamt			**1680€**

Tabelle 3: Hardwarekosten

$$\frac{1680\,€}{3\text{ Jahre}} = 560/\text{Jahr} \tag{5}$$

$$\frac{560\,€/\text{Jahre}}{1840\text{ h/Jahr}} \approx 0.30\,€/\text{h} \tag{6}$$

Damit ergab sich ein Stundensatz von 0.3 € für die Hardware.
Sonstige wesentliche Kosten neben den Hardwarekosten sind natürlich auch Arbeitsplatz, Möblierung, Strom, Internet und Lizenzen, welche einen Großteil der Ressourcenkosten ausmachen. Durch grobe Hochrechnung dieser ergaben sich insgesamt Ressourcenkosten (ohne Personal) von ca. 10 € als Stundensatz.

Vorgang	Mitarbeiter	Azubi	Zeit	Personal	Ressourcen	Gesamt
Anforderungsaufnahme	2	1	2 h	114 €	60 €	124 €
Ressourcen Entwicklung		1	39 h	273 €	390 €	663 €
Testphase		1	3 h	21 €	30 €	51 €
Code Reviews	2	1	3 h	171 €	90 €	281 €
Fazit und Dokumentation		1	23 h	161 €	230 €	391 €
Gesamt						**1510 €**

Tabelle 4: Projektkosten

Wie in Tabelle 4 ermittelt, betrugen die gesamten Projektkosten 1510 €.

3.2.2 Amortisationsdauer

Im folgenden Abschnitt wird gezeigt, wann sich die Entwicklung aus wirtschaftlicher Sicht für die Camelot ITLab GmbH und für den Kunden lohnt. Die Amortisationsdauer wird berechnet, indem man die Produktionskosten bzw. Anschaffungskosten durch die Gewinne bzw. Kostenersparnis, die aufgrund dieses Produktes entstanden sind, dividiert.

Camelot ITLab GmbH

Für dieses Tool wird ein Preis von etwa 500€ angesetzt. Der Break-Even-Point (Gewinnschwelle) liegt also bei

$$\frac{1510\,€}{500\,€} \approx 3 \text{ verkauften Einheiten} \tag{7}$$

Kunde

Hier lassen sich die Gewinne nicht so leicht erfassen, da zum einen die Fehleranfälligkeit der alten Lösungen (Kommentare per Email verschicken, Zettelwirtschaft usw.) deutlich verringert werden, zum anderen erspart das Tool dem Planer, welcher die Aufträge anlegt, deutlich Zeit, da dieser nicht noch ein weiteres Programm benötigt, um die Kommentare einzutragen und an seine Kollegen zu verteilen. Des Weiteren ermöglicht es den Mitarbeitern, an der Ressource (Maschine) Kommentare direkt in der Transaktion neben den Aufträgen zu sehen. In der folgenden Beispielrechnung wurden drei Szenarien einmal durchgerechnet.

Beispielrechnung Es wird davon ausgegangen, dass ein Planer jeden Tag ca. eine Stunde damit beschäftigt ist, Aufträge zu kommentieren. Dieses Tool bietet eine Zeitersparnis von ca. 50%, da die Kommentare direkt in der Transaktion verfasst werden können. Dies bedeutet eine Zeiteinsparung von 30 Minuten pro Tag und pro Planer. Bei etwa 230 Arbeitstagen pro Jahr ergibt sich eine gesamte Zeiteinsparung von

$$230 \text{ Tage/Jahr} \cdot 30 \text{ min/Tag} = 6900 \text{ min/Jahr} = 115 \text{ h/Jahr} \tag{8}$$

Das Gehalt eines Produktionsplaners beträgt etwa 14 € die Stunde. Die tatsächlichen Kosten, incl. Nebenkosten, die für die Firma anfallen, werden auf ca. 20€ geschätzt. Dadurch ergibt sich eine jährliche Einsparung von

$$115\text{h} \cdot (20)\,€/\text{h} = 2300\,€ \tag{9}$$

Je nach Firmengröße und Anzahl der Planer variiert die Amortisationszeit stark. Daher wurden Anhand dreier Beispiele die Zeit berechnet.

Die Amortisationszeit für eine kleine Firma mit 2 Planern beträgt: $\frac{500\,€}{2 \cdot 2300€\,€/\text{Jahr}} \approx 0,10$ Jahre $\approx$ 5 Wochen.

Die Amortisationszeit für eine mittelgroße Firma mit 50 Planern beträgt $\frac{500\,€}{50 \cdot 2300€\,€/\text{Jahr}} \approx 0,004$ Jahre $\approx$ $1,5$ Tage.

Die Amortisationszeit für eine große Firma mit 200 Planern beträgt $\frac{500\,€}{200 \cdot 2300€\,€/\text{Jahr}} \approx 0,001$ Jahre $\approx$ 9 Stunden.

Man erkennt, dass das Projekt sowohl für die Camelot ITLab GmbH als auch für den Kunden wirtschaftlich Sinnvoll ist.

3.3 Nutzwertanalyse

In der folgenden Tabelle 5 werden noch einige weitere Punkte aufgeführt, welche abseits der primären finanziellen Aspekte für diese Produkt bzw. die alten Verfahren sprechen. Für Kriterien, die in direktem Zusammenhang mit der Produktion und den erzielten Ergebnissen stehen, wird eine hohe Gewichtung angesetzt. Kriterien wie etwa Komfort, welche abhängig von persönliche Präferenzen des Planers sind, werden eher niedrig eingestuft.

Kriterien	Max. Punkte	Wichtung	OC Tool	Per Mail	Per Zettel
Geschwindigkeit	10	4	8	4	6
Transport	10	4	10	10	1
Zuverlässigkeit	10	4	10	8	2
Komfort	10	1	8	3	2
Gestaltungsmöglichkeit	10	1.5	2	8	8
Gesamt			123	103	50

Tabelle 5: Nutzwertanalyse

Sowie die Wirtschaftlichkeitsanalyse als auch die Nutzwertanalyse haben gezeigt wie sinnvoll und hilfreich diese Projekt ist.

3.4 Anwendungsfälle

Während der Analysephase wurden einige der typischen Anwendungsfälle, die von den umzusetzenden Programmen und Erweiterungen abgedeckt werden sollen, mittels eines Use-Case-Diagramms zusammengetragen, um einen groben Überblick über diese zu erhalten. Dieses Diagramm befindet sich im Anhang A.5: Use-Case-Diagramm auf Seite v.

3.5 Qualitätsanforderungen

Wie bei jedem anderen Projekt der Camelot ITLab GmbH gelten auch hier die Programmierrichtlinien des Unternehmens. Eine Auszug mit den für dieses Projekt wichtigen Punkten findet sich im Anhang A.9: Programmierrichtlinien auf Seite xix. Außerdem muss es für den Administration Screen eine Eingabeüberprüfung geben, die verhindert, dass der User Keyfelder bzw. Felder, die in der Datenbanktabelle nicht vorhanden sind, angibt.

3.6 Lastenheft/Fachkonzept

Das Lastenheft beschreibt die vom "Auftraggeber festgelegte Gesamtheit der Forderungen an die Lieferungen und Leistungen eines Auftragnehmers innerhalb eines Auftrags"(Wiki.Induux-Website [2018]). Das Lastenheft, welches im Bezug auf dieses Projekt entstanden ist, befindet sich im Anhang A.3: Lastenheft (Auszug) auf Seite iii.

3.7 Zwischenstand

Tabelle 6 zeigt den Zwischenstand nach der Analysephase.

Vorgang	Geplant	Tatsächlich	Differenz
1. Analyse des Ist-Zustands	1.5 h	1.5 h	
2.„Make or buy"-Entscheidung und Wirtschaftlichkeitsanalyse	1 h	1 h	
3. Erstellung des „Use-Case"-Diagramms	0.5 h	0.5 h	
4. Erstellung des Lastenhefts	3 h	3 h	
Differenzgesamt			0 h

Tabelle 6: Zwischenstand nach der Analysephase

4 Entwurfsphase

4.1 Zielplattform

Das Abschlussprojekt sollte wie bereits im Projektziel beschrieben eine Erweiterung zu bereits vorhandenen Transaktionen darstellen und eigene Programme zur Verwaltung und Massenpflege der Kommentare erbringen. Als Programmiersprache wurde ABAP verwendet, eine eigens von der SAP entwickelte Programmiersprache, die in ihrer Grundstruktur der Sprache COBOL ähnelt.

4.2 Architekturdesign

Die Programme im APO und ECC wurden nach dem Model View Controller (MVC) Konzept programmiert. Allerdings habe ich in diesem Fall kein Model benötigt. Es gibt in jedem System jeweils eine Graphical User Interface (GUI) Klasse, welche für die visuelle Darstellung und die Reaktion auf Benutzerinteraktionen zuständig ist. Diese Klasse besitzt für jeden Screen eine Member Struktur, die die anzuzeigenden Daten hält und jeweils eine Process Before Output (PBO) und Process After Input (PAI) Methode. Die jeweiligen Screens wurden in einer Funktionsgruppe definiert und mithilfe des SAP Screen Painters gestaltet. Außerdem gibt es jeweils eine Controller Klasse, welche für die gesamte Logik zuständig ist. Die Startpunkte der Programme sind jeweils ein Report, welcher entweder einen Selektionsbildschirm hat oder direkt über ein Funktionsmodul, welches in der Funktionsgruppe definiert ist, den gewünschten Screen aufruft, da aus einem Report direkt kein Screen aufgerufen werden kann. Das PBO und das PAI in der Funktionsgruppe sind dynamisch agierende Module, welche dann auf die jeweilige Methode der GUI Klasse weiterleiten.

4.3 Entwurf des Userinterface

Um die Anwendungen möglichst benutzerfreundlich zu gestalten, wurde im Vorfeld klar strukturiert, auf welchem Screen der User welche Informationen angezeigt bekommen soll. Außerdem wurden die möglichen Selektionskriterien vorab geplant, damit später keine Zeit mit der Erstellung von unnötigen Datenstrukturen verbraucht wird. Es wird später im ECC einen Screen mit dem Namen Administration geben, auf welchem der Planer ein Feld der Datenbank Tabelle AUFK angeben kann, in welchem dann der Kommentar gespeichert wird. Hier wurde darauf geachtet, dass der Code leicht zu erweitern ist, da später auch noch weitere Tabellen und Felder ausgewählt werden sollen können wie z.B. die PLAF, in welcher Planaufträge gespeichert sind. Auf dem Hauptscreen des Programms werden die Auftragsdaten und die zugehörigen Kommentare in einem ABAP List View (ALV) dargestellt. Im APO wird es denselben Hauptscreen auch geben, hier fällt allerdings der Administration Screen weg, da hier alle Kommentare unabhängig von der Kategorie in der Datenbank /SAPAPO/ORDFLDS gespeichert werden.

4.4 Datenmodell

Im ECC werden zwei Datenbank Tabellen erstellt. Zum einen die /CAMELOT/OC_COMT. Der Aufbau der Datenbanken und die Erweiterungen werden in den folgenden Unified Modeling Language (UML)s dargestellt.

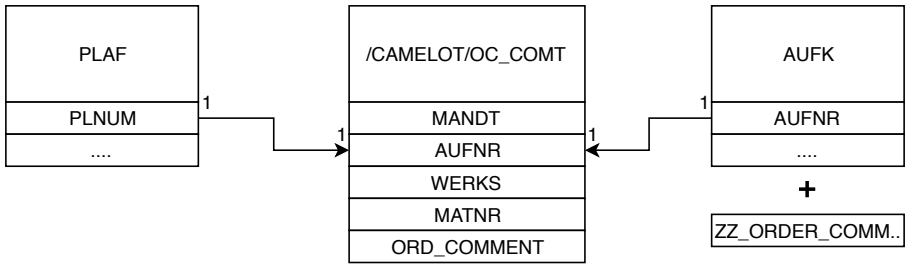


Abbildung 1: Vereinfachtes Datenbank-Model der /CAMELOT/OC_COMT

Die Tabelle AUFK wird mittels einem Custom Include um ein Feld ZZ_ORDER_COMMENT erweitert. Dieses wird für die COR Erweiterung gebraucht. Außerdem wird eine Datenbank Tabelle /CAMELOT/OC_SETT erstellt, wie in dem folgenden UML beschrieben. In dieser Datenbank wird vorerst lediglich ein Feldname der AUFK gespeichert. Später muss diese Tabelle dann noch um einen Tabellenname erweitert werden.

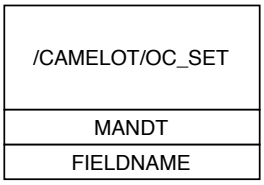


Abbildung 2: Vereinfachtes Datenbank-Modell der /CAMELOT/OC_SET

Im APO wird keine extra Datenbank zum Speichern der Kommentare benötigt, da hier, wie bereits oben erwähnt,direkt die /SAPAPO/ORDFLDS per Custom Include um ein FELD ORDER_COMMENT erweitert und für alle Aufträge genutzt werden kann.

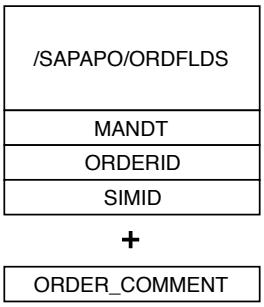


Abbildung 3: Vereinfachtes Datenbank-Modell der /SAPAPO/ORDFLDS

4.5 Maßnahmen zur Qualitätssicherung

Um die hohen Qualitätsanforderungen der Camelot ITLab GmbH zu gewährleisten und die Qualitätsanforderungen des Projekts zu sichern, wurden während der laufenden Entwicklung nach jedem Iterationsschritt die neu eingebauten Funktionalitäten getestet. Außerdem gab es mehrere Code Reviews in denen andere Entwickler sich den Code anschauten und nach Schwachstellen suchten und ggf. Verbesserungsvorschläge einbrachten. Alle Tests wurden manuell durchgeführt.

4.6 Pflichtenheft/Datenverarbeitungskonzept

Ein Beispiel für das auf dem Lastenheft (siehe Kapitel 3.6: Lastenheft/Fachkonzept) aufbauende Pflichtenheft ist im Anhang A.6: Pflichtenheft (Auszug) auf Seite v zu finden.

4.7 Zwischenstand

Tabelle 7 zeigt den Zwischenstand nach der Entwurfsphase.

Vorgang	Geplant	Tatsächlich	Differenz
1. Iterationsplanung	2.5 h	2 h	+0.5 h
2. Erstellung des Pflichtenhefts	2 h	3 h	-1 h
3. Erstellung der Datenbank Modelle	1 h	1 h	
Differenzgesamt			-0.5 h

Tabelle 7: Zwischenstand nach der Entwurfsphase

5 Implementierungsphase

5.1 Iterationsplanung

Bevor mit der eigentlichen Implementierung begonnen wurde, wurde zuerst ein Interationsplan erstellt. In ihm werden die Iterationsschritte und deren Reihenfolge definiert. Innerhalb einer Iteration wird die zuvor definierte Funktionalität eingebaut. Der erstellte Iterationsplan befindet sich im Anhang Anhang A.4: Iterationsplan auf Seite iv.

5.2 Implementierung der Datenstruktur ECC

Zuerst wurden alle Dictionary-Objekte, welche im ECC gebraucht werden, erstellt. Eine Vollständige Liste aller Tabellen Typen, Strukturen (einer Art mehrdimensionaler Array), Datenelemente und Domänen können dem Anhang entnommen werden. Außerdem wurden die Datenbank Tabellen wie in Kapitel 4.4: Datenmodell beschrieben implementiert. Die AUFK wurde mittels einem Custom Include um das Feld ZZ_ORDER_COMMENT erweitert.

5.3 Implementierung der Benutzeroberfläche ECC

In der GUI Funktionsgruppe wurden jeweils ein Screen für das Maintenance Programm (Screen 0100) und ein Screen für das Administration Programm (Screen 0500) mithilfe des Screen Panters angelegt und gestaltet. Der Screen 0100 enthält lediglich einen Custom Container, in welchem dann später das ALV angezeigt wird. Der Screen 0500 hat zum jetzigen Zeitpunkt nur ein Label und ein Eingabefeld, dessen Element sich in der Member Struktur des Screens der GUI Klasse befindet. Das PBO und das PAI der Funktionsgruppe wurde dynamisch programmiert. Das bedeutet, dass je nach Screen die zugehörige Methode in der GUI Klasse aufgerufen wird. Die User Befehle (OK Code) werden in ein Member Feld der GUI Klasse geschrieben. Außerdem wurde für das Maintenance Programm ein extra Selektionsbildschirm angelegt, in welchem man gewisse Parameter wie Benutzer, Auftragsnummer usw. eingeben kann und somit die dargestellten Aufträge gefiltert werden können. Screenshots der Anwendung befinden sich unter Anhang A.8: Screenshots der Anwendung auf Seite ix.

Screenshots der Anwendung in der Entwicklungsphase mit Dummy-Daten befinden sich im Anhang A.8: Screenshots der Anwendung auf Seite ix.

5.4 Implementierung PBO und PAI

Jeder Screen der Funktionsgruppe hat seine eigene PBO und PAI Methode in der GUI Klasse. Im PBO werden die Daten geladen, bevor sie dann auf dem Screen angezeigt werden. Dies passiert, indem die jeweilige Controller Methode aufgerufen wird. Die Daten werden allerdings nicht jedes Mal neu geladen wenn das PBO aufgerufen wird, sondern nur, wenn die ALV Struktur, welche in der

Member Struktur des jeweiligen Screens ist, Initial also “leer“ ist. Dadurch wird verhindert, dass zu viele Datenbankzugriffe erfolgen, da jedes Mal die Daten neu geladen werden. Nichtsdestotrotz kann eine Refresh einleiten werden, indem einfach die ALV Struktur geleert wird. Im PAI werden all User Befehle (OK-Code) abgefangen, welche vom Screen geworfen werden und dann die jeweilige Action ausgeführt. Den OK-Code des jeweiligen Buttons wird im Screen-Painter bzw. im Status des jeweiligen Screens definiert.

5.5 Implementierung der Geschäftslogik ECC

Die eigentliche Geschäftslogik und die Datenbankzugriffe finden alle im Controller statt. Hier wurden mehrere Methoden implementiert, welche die gesamten Aufträge, die derzeit unterstützt werden, (Prozessaufträge, Produktionsaufträge aus der AUFK mit join auf die AFKO für Material Infos und Planaufträge aus der PLAF) laden. Außerdem wurde die Speicherlogik implementiert. In der Member Struktur des Screens gibt es zwei Tabellen, eine für die Daten, die dann tatsächlich im ALV angezeigt werden und eine andere, in welcher immer die originalen Daten seit dem letzten Speichern enthalten sind. Beim Speichern werden nun diese zwei Tabellen verglichen und so alle Aufträge, welche sich nicht geändert haben, aussortiert, sodass die Speicherlogik nur auf die tatsächlich modifizierten oder erstellten Aufträge angewendet wird. Diese Aufträge werden dann mithilfe eines RFC ins APO System transferiert. (siehe Kapitel 5.9: Implementierung der Geschäftslogik APO)

5.6 Implementierung der COR Erweiterung

Um die COR Transaktion (1-3) zu erweitern, musste zunächst einmal ein passender Erweiterungspunkt gefunden werden, welcher zum einen, einen Screen Exit und zum anderen, zwei Functions Exits vor und nach dem Laden der Daten hat. Verwendet wurde das Enhancement PPCO0020. Diese bietet alle Komponenten, die zum Anzeigen der Daten benötigt wurden. Zuerst wurde das Screen Exit erstellt und mittels des Screen Painters ein simpler Screen mit einem Label und einem Eingabefeld erstellt und aktiviert. Dann wurde das PBO Modul um eine Methode erweitert, sodass das Eingabefeld in der COR3 deaktiviert wird, da diese Transaktion nur zum Anzeigen ist. Außerdem wurde in dem Top Include eine globale Struktur vom Typ AUFK angelegt, dessen ZZ_ORDER_COMMENT Feld hinter dem Eingabe Feld liegt. In dem ersten Function Exit, wird diese Struktur dann gefüllt. Da der Planer aber auch die Möglichkeit haben soll, Kommentare von Prozessaufträgen zu ändern bzw. beim Anlegen eines Auftrags anzugeben, wurde noch ein weiterer Funktionsbaustein benötigt, da der oben genannte, keinen Function Exit für das Speichern hat, von wo aus unsere eigene Speicherlogik aufgerufen werden kann. Hierzu wurde der Erweiterungspunkt PPCO0007 gewählt. Dieser Erweiterungspunkt liefert einen Function Exit aus welchem dann eine statische Methode aus dem Controller aufgerufen wird, damit der geänderte Kommentar nicht nur in der AUFK sondern auch in der /CAMELOT/OC_COMT gespeichert wird und von wo außerdem der Remote Function Call (RFC) ausgeführt wird, damit der Kommentar ins APO transferiert wird. Näheres (siehe Kapitel 5.9: Implementierung der Geschäftslogik APO)

5.7 Implementierung der Datenstruktur APO

Nachdem im ECC System nun alles Implementiert worden war, musste nun ein Großteil der selben Datenstrukturen auch im APO angelegt werden. Eine Liste aller Dictionary-Objekte kann dem Anhang entnommen werden. Des Weiteren wurde die /SAPAPO/ORDFLDS Datenbank Tabelle um ein Feld, ORDER_COMMENT, erweitert. Im APO wird keine extra Datenbank wie im ECC benötigt, da für alle Aufträge die /SAPAPO/ORDFLDS Tabelle benutzt werden kann.

Screenshots der Anwendung in der Entwicklungsphase mit Dummy-Daten befinden sich im Anhang A.8: Screenshots der Anwendung auf Seite ix.

5.8 Implementierung der Benutzeroberfläche APO

Ähnlich wie im ECC musste ein Screen(0100) erstellt werden, auf welchem das ALV mit den Auftragsdaten und dem Kommentar angezeigt wird. Der zweite Screen, der im ECC angelegt wurde, wird im APO nicht benötigt, da automatisch alle Kommentaren in der /SAPAPO/ORDFLDS im Feld ORDER_COMMENT gespeichert werden. Der Screen 0100 ist genauso aufgebaut wie im ECC, nur der Selektionsbildschirm fällt zum jetzigen Zeitpunkt deutlich kleiner aus, da man nur nach Auftragsnummern filtern kann.

5.9 Implementierung der Geschäftslogik APO

Wie im ECC findet auch im APO die gesamte Geschäftlogik im Controller statt. Neben denselben Klassen,die auch im ECC vorhanden sind(Constants, Controller, GUI, ALV) gibt es eine extra Klasse mit dem Namen /CAMELOT/CL_OC_RRP, welche für den BAdI und die implizite Erweiterung der RRP3 genutzt wird. Außerdem gibt es neben der Funktionsgruppe, welche für das Hauptprogramm genutzt wird, eine weitere Funktionsgruppe /CAMELOT/OC_COMMENT mit dem Funktionsmodul /CAMELOT/OC_COMMENT_SYNC welche Remote-Enabled ist. Das bedeutet sie kann aus einem anderen System heraus aufgerufen werden. Diese Methode hat als Importparamter eine Tabelle mit dem Tabellentyp /CAMELOT/OC-_ORD_COMMENT_RFC_T (siehe Anhang A.7: Dictionary-Objekte auf Seite vii). Diese Funktion dient zum Transport der Kommentare vom ECC ins APO. Die Struktur der Tabelle beinhaltet nur die Auftragsnummer und den Kommentar, um den Traffic möglichst gering zu halten. Die Methode selber sucht mithilfe eines SAP Funktionsbausteins /SAPAPO/DM_ORDER-_GET_ORDID die richtige Orderid aus dem Livecache zu der gegebenen Auftragsnummer. Mithilfe der Orderid und dem Kommentar wird dann die /SAPAPO/ORDFLDS Tabelle geupdatet.

5.10 Implementierung der RRP3 Erweiterung

Um die RRP3 zu erweitern wurde nicht wie bei der COR1-3 Customer-Exits sondern ein BAdI und eine implizite Erweiterung verwendet. Der Unterschied zwischen einem Customer-Exits und einem BAdI ist, dass der BAdI die neuere objektorientierte Variante eines Customer-Exits ist. Statt in Funktionsmodule Code einzufügen wird eine vordefinierte Klasse implementiert, deren Methoden wie bei den alten Customer-Exits zu einem bestimmten Zeitpunkt aufgerufen werden. Für diese RRP3 Erweiterung wurden der BAdI /SAPAPO/RRP_IO_COL benötigt.Mit diesem war es möglich, an den Feldkatalog der RRP3 das Feld Order_Comment anzufügen und dieses dann auch mittels einer zweiten Methode mit den richtigen Kommentaren zu füllen. Außerdem musste eine Implicites Enhancement angelegt werden, da es mithilfe von Customer-Exits oder BAdIs nicht möglich war, die t-style Tabelle der einzelnen Elemente in der ALV Tabelle zu ändern, was allerding absolut notwendig ist, um z.B. den Kommentar editierbar zu machen. Die jeweiligen Methoden des BAdIs und der impliziten Erweiterung rufen eine statische Methode in der RRP3 Klasse auf, welche genau den selben Namen hat wie die eigentliche Methode des BAdIs, um einen besseren Überblick zu verschaffen und sie später gut bei Kunden implementieren zu können. Diese Methoden holen sich dann die Instance des Controllers bzw. eine wird erstellt und rufen dann die jeweiligen Controller Methoden auf.

5.11 Zwischenstand

Tabelle 8 zeigt den Zwischenstand nach der Implementierungsphase.

Vorgang	Geplant	Tatsächlich	Differenz
1. Implementierung der Datenbanken und Dictionary-Objekten im ECC	1 h	1 h	+1 h
2. Implementierung der Klassen im ECC	9 h	9 h	+ 0.1 h
3. Implementieren der COR Erweiterung	2 h	2 h	
4. Implementierung der Datenbank und Dictionary-Objekten im APO	1 h	1 h	+1 h
5. Implementierung der Klassen im APO	5.5 h	6 h	-0.5 h
6. Implementieren der RRP3 Erweiterung	3 h	3 h	
Differenzgesamt			+0.5 h

Tabelle 8: Zwischenstand nach der Implementierungsphase

6 Qualitätssicherung

6.1 Code Reviews während des Projekts

Während des Projekts, in der Mitte und am Ende gab es zwei Code Reviews welche jeweils 1 Stunde lang gingen und wo zwei Mitarbeiter der Entwicklungsabteilung gemeinsam mit dem Auszubildenden über den Code geschaut haben und mögliche Schwachstellen diskutiert und verbessert haben. Außerdem wurden wertvolle Tipps gegeben wie das Programm noch besser umgesetzt werden kann.

6.2 Manuelle Tests

Während der Entwicklungsphase wurde nach jedem Iterationsschritt die neu implementierte Funktionalität getestet und gegebenenfalls verbessert. Diese Tests haben länger gedauert als erwartet, jedoch konnte der Zeitverlust durch weniger Zeit für die Bug Fixes kompensiert werden, sodass diese Phase nicht länger als die Veranschlagten 6 Stunden gedauert hat. Außerdem wurde nach Abschluss der Entwicklung noch einmal der komplette Funktionsumfang der Anwendung getestet.

6.3 Bug Fixing

Alle Bugs, die während der Entwicklung aufgefallen sind, wurden immer direkt korrigiert bzw. schriftlich vermerkt, sodass keine Fehler in Vergessenheit gerieten.

7 Fazit

7.1 Soll-/Ist-Vergleich

Die Zeit von 70 Stunden und der in Abschnitt 2.1 (Projektphasen) erstellte Projektplan wurden insgesamt eingehalten auch wenn es bei den einzelnen Projektphasen zu Verschiebungen gekommen ist. Alle festgelegten Ziele aus dem Pflichtenheft wurden erfolgreich Umgesetzt. Die Implementierung im ECC System hat etwas länger gedauert als zuerst angenommen, dafür konnte ein Großteil der Logik ins APO System kopiert und hier einige Zeit gespart werden. In der Tabelle 9: Soll-/Ist-Vergleich der Projektphasen Zeitplanung wird die tatsächlich benötigte Zeit gegenüber die geplante Zeit gestellt und verglichen.

Phase	Geplant	Tatsächlich	Differenz
Anforderungsaufnahme	2 h	2 h	
Planung	4 h	4 h	
Analysephase	6 h	6 h	
Entwurfsphase	6 h	5,5 h	-0,5 h
Implementierungsphase	23 h	23.5 h	+0,5 h
Testphase/Qualitätsicherung	6 h	6 h	
Fazit	1 h	1 h	
Dokumentationsphase	20 h	20 h	
Gesamt	70 h	70 h	

Tabelle 9: Soll-/Ist-Vergleich der Projektphasen Zeitplanung

7.2 Ausblick

Alle im Lastenheft definierten Anforderungen konnten erfüllt werden.Dennoch möchte ich hier einen kleinen Ausblick auf mögliche weitere Feature geben, die während der Entwicklung aufgefallen und als praktisch erachtet wurden. Zum einen gibt es noch weitere Transaktionen, die erweiter werden können wie z.B. die MD04, um den Planer noch mehr Komfort zu bieten. Außerdem sollte man in einer späteren Version im ECC im Administration Screen nicht nur einen Feldnamen angeben können, sondern auch die Datenbanktabelle bzw. mehrere Einträge vornehmen können. Außerdem sollte in absehbarer Zukunft ein Lösch Programm geschrieben werden, das die Aufträge, welche sich im ECC System in der /CAMELOT/OC_COMT Datenbanktabelle befinden überprüft und Aufträge welche nicht mehr existieren aus der Tabelle löscht. Aufgrund des in Abschnitt Architekturdesign beschriebenen Aufbaus des Programms lassen sich Änderungen und Anpassungen sehr einfach vornehmen. Außerdem erhöht dies die Wartbarkeit der Programme, da alle Mitarbeiter des Camelot ITLabs sich besondern mit dieser Struktur auskennen.

Quellenverzeichnis

Fachinformatiker-Website 2018

Fachinformatiker-Website: *Fachinformatiker-anwendungsentwickler*. Version: 2018. https://fachinformatiker-anwendungsentwicklung.net/beispiele-fuer-abschlussprojekte/, Abruf: 04.12.2018

Horst Keller 2006

Horst Keller, Sascha K.: *ABAP Objects*. Galileo Press, 2006

Ptak und Smith 2016

Ptak ; Smith: *DDMRP*. Industrial Press, Inc, 2016

SAP.archive 2018

SAP.archive: *SAP.Archive-Website*. Version: 2018. https://archive.sap.com, Abruf: 04.12.2018

Stackexchange-Website 2018

Stackexchange-Website: *Tex.stackexchange*. Version: 2018. https://tex.stackexchange.com/test, Abruf: 04.12.2018

Wiki.Induux-Website 2018

Wiki.Induux-Website: *Wiki.Induux*. Version: 2018. https://wiki.induux.de/Lastenheft, Abruf: 04.12.2018

wikipedia.de 2018

wikipedia.de: *Wikipedia-Website*. Version: 2018. https://de.wikipedia.org/wiki/ABAP/, Abruf: 04.12.2018

A Anhang

A.1 Detaillierte Zeitplanung

Anforderungsaufnahme			**2 h**
Planung			**4 h**
Analysephase			**6 h**
1. Analyse des Ist-Zustands		1,5 h	
1.1. Fachgespräch mit dem Ausbilder		1 h	
1.2. Prozessanalyse		0,5 h	
2. „Make or buy“-Entscheidung und Wirtschaftlichkeitsanalyse		1 h	
3. Erstellen eines „Use-Case“-Diagramms		0.5 h	
4. Erstellung des Lastenhefts mit dem Ausbilder		3 h	
Entwurfsphase			**6 h**
1. Iterationsplanung		2 h	
2. Erstellen des Pflichtenhefts		3 h	
3. Erstellung der Datenbank Modelle		1 h	
Implementierungsphase			**23 h**
1. ECC Implementierung		12 h	
1.1. Implementierung der Datenbanken und Dictionary-Objekten		1 h	
1.2. Funktionsgruppe erstellen		1 h	
1.2.1. Screens erstellen	0.5 h		
1.2.2. Stati und Titel erstellen	0.5 h		
1.3. Implementierung der Klassen		8 h	
1.3.1 Implementierung der GUI Klasse, PBO und PAI	2 h		
1.3.2. Ladelogik implementieren	2 h		
1.3.3. Speicherlogik implementieren	4 h		
1.4. COR Erweiterung implementieren		2 h	
2. APO Implementierung		11 h	
2.1. Implementierung der Datenbanken und Dictionary-Objekten		1h	
2.2. Funktionsgruppe erstellen		1h	
2.2.1. Screens erstellen	0.5 h		
2.2.2. Stati und Titel erstellen	0.5 h		
2.3 Implementierung der Klassen		6 h	
2.3.1 Implementierung der GUI Klasse, PBO und PAI	1 h		
2.3.2 Ladelogik implementieren	1 h		
2.3.3 Speicherlogik implementieren	4 h		
2.4. RRP3 Erweiterung implementieren		3 h	
Testphase/Qualitätsicherung			**6 h**
1. Code Reviews		2 h	
2. Manuelle Tests		1 h	
3. Bug Fixing		3 h	
Fazit			**3 h**
1. Soll-/Ist-Vergleich		1 h	
2. Ausblick		2 h	
Dokumentationsphase			**20 h**
1. Erstellen der Projektdokumentation		20 h	
Gesamt			**70 h**

A.2 Ressourcen Übersicht

Typ	Bezeichnung	Rolle
Personal		
Thomas Pöhlmann	Entwickler und Autor	Projektausführung und Dokumentation
Florian von der Weth	Senior Consultant	Auftraggeber
Florian P.	Junior Consultant	CodeReview
Julian P.	Consultant	CodeReview
Hardware		
	Lenovo ThinkPad T450s	Arbeits und Testgerät
	2 * Dell 24 P2419H 61	Hauptbildschirme
Software		
Betriebssystem	Microsoft Windows 10	Arbeits-/Test-Betriebssystem
Entwicklungs-Tools	ABAP Development Workbench	Code Editor
Dokumentation	TexStudio	Latex Editor
	TexWorks	Editor für Bib files
	Microsoft Excel	Tabellen Editor
Websites	http://draw.io	Grafik Editor
	http://excel2latex.com/	Excel -> Latex Konvertierer
	https://gitlab.com/	Versionsverwaltung
Sonstiges	GitExtension	UI für GIT

A.3 Lastenheft (Auszug)

Die Anwendungen müssen folgende Anforderungen erfüllen:

1. Darstellung der Daten
   1.1. Die Anwendung muss die Funktion bieten Aufträge nach vorgegebenen Kriterien filter zu können.
   1.2. Die Aufträge müssen in einer für den Planer übersichtlichen Umgebung dargestellt werden.
   1.3. Der Planer soll möglichst einfach und schnell Kommentare schreiben und anzeigen können.
   1.4. Der Planer soll zudem auch in den gewohnten Standardanwendungen Kommentare verfassen und lesen können
2. Weitere Anforderungen
   2.1. Die Anwendungen sollen sauber und übersichtlich programmiert werden, sodass spätere Erweiterungen leicht durchzuführen sind.
   2.2. Möglichst wenig Code soll in Userexits oder sonstigen Erweiterungspunkten geschrieben werden, da dieser immer per Hand beim Kunden eingefügt werden müssen.

A.4 Iterationsplan

1. ECE Entwicklung
   1.1. Erstellung der Dictionary-Objekte
   1.2. Erstellung der Datenbanken
   1.3. Erstellung der Klassen ALV, GUI, Controller und Constants
   1.4. GUI Funktionsgruppe erstellen
   1.5. Maintenance Screen erstellen
   1.6. Administration Screen erstellen
   1.7. AUFK erweitern und Speicher Logik einbauen
   1.8. COR Programm erweitern
2. SCE Entwicklung
   2.1. Erstellung der Da Dictionary-Objekte
   2.2. Erweiterung der /SAPAPO/ORDFLDS
   2.3. RFC fähiges Funktionsmodul erstellen mit Speicherlogik zur Übertragung der Kommentare aus dem ECC ins APO
3. ECE Entwicklung
   3.1. In den vorhandenen Speicherlogiken den RFC einbauen
4. SCE Entwicklung
   4.1. Erstellung der Klassen ALV, GUI, Controller und Constants
   4.2. GUI Funktionsgruppe erstellen
   4.3. Maintenance Screen erstellen
   4.4. RRP3 Enhancement implementieren

A.5 Use-Case-Diagramm

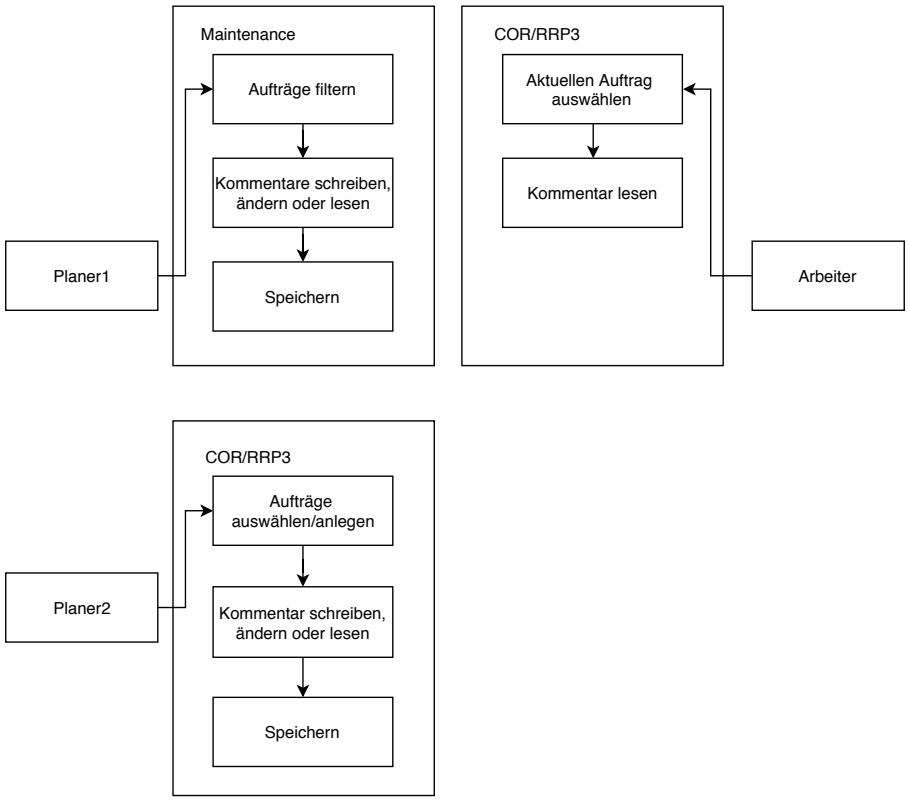


Abbildung 4: Use-Case-Diagramm

A.6 Pflichtenheft (Auszug)

Zielbestimmung

1. Musskriterien
   1.1. ECC System
   - Ein neues Package muss im ECC angelegt werden mit dem Namen /CAMELOT/OC.
   - Die AUFK Datenbanktabelle muss um ein Feld mit dem Namen ZZ/_ORDER/_COMMENT erweitert werden
   - Ein extra Programm, welches eine gefilterte Auswahl an Prozess-, Plan- und Produktions-Aufträgen anzeigt, muss erstellt werden.
   - Ebenfalls sollen hier jeweils die Kommentare für die Aufträge angezeigt und vom User geändert werden können.
   - Weiterhin soll dieses Programm massenänderungsfähig sein.
   - Außerdem soll es ein zweites Programm geben, in die der User ein Tabellenfeld der AUFK Datenbanktabelle angeben kann, welches dann für die Kommentare genutzt wird.

- Mittels Input Checks soll verhindert werden, dass der User ein Feld angibt, das es in der AUFK Tabelle nicht gibt
- Zum Schluss soll die COR1-3 erweitert werden.
  - Ein neues Package muss angelegt werden mit dem Namen ZPP, in welches dann die COR Erweiterung kommt.
  - Mittels der cmod Transaktion muss ein neues Projekt mit dem Namen Z_COR angelegt werden und zwei Erweiterungspunkte hinzugefügt werden (PPCO0007 und PPCO0020).

1.2. APO System.

- Es muss ebenfalls ein neues Package angelegt werden /CAMELOT/OC
- Die /SAPAPO/ORDFLDS muss um ein Feld mit dem Namen Order_Comment erweitert werden.
- Ebenso wie im ECC System soll es ein Programm geben, welches mittels vom User eingegebener Order Number einen Auftrag mit Kommentar anzeigt, der vom User geändert werden kann.
- Desweiteren soll die RRP3 erweitert werden.
  - Es muss ein neues Package mit dem Namen ZSCM angelegt werden.
  - Mittels der se18 muss ein BAdI in dieses Package implementiert werden.
  - Die Logik muss programmiert werden.
  - Eine implizite Erweiterung muss angelegt werden, um die T_style Tabelle zu manipulieren.

A.7 Dictionary-Objekte

A.7.1 Tabellentypen

Name	Zeilentyp
/camelot/oc_aufk_T	aufk
/camelot/oc_comt_t	/camelot/oc_comt
/camelot/oc_dats_range_t	/camelot/oc_dats_range_s
/camelot/oc_gui_0100_alv_t	/camelot/oc_gui_0100_alv_s
/camelot/oc_matnr_range_t	/camelot/oc_matnr_range_s
/camelot/oc_ordnr_range_t	/camelot/oc_ordnr_range_s
/camelot/oc_ord_comment_RFC_T	/camelot/oc_ord_comment_rfc
/camelot/oc_PLANT_RANGE_T	/camelot/oc_plant_range_s
/camelot/oc_plnum_range_t	/camelot/oc_plnum_range_s
/camelot/oc_screen_t	screen
/camelot/oc_uname_range_t	/camelot/oc_uname_range_s

A.7.2 Strukturen

Name	Componenten	Typen
/camelot/oc_dats_range_s	sign	ddsign
	option	ddoption
	low	dats
	high	dats
/camelot/oc_gui_0100_alv_s	aufnr	aufnr
	matnr	matnr
	werks	werks_d
	created_by	/camelot/oc_created_on
	created_on	/camelot/oc_created_by
	start_date	pm_ordgstrp
	end_date	co_gltrp
	ord_comment	/camelot/oc_ord_comment
	t_style	lvc_t_style
	category	char2
/camelot/oc_gui_maint_sel_s	s_selection	/camelot/oc_gui_maint_sel_s
	t_alv_orders	/camelot/oc_gui_0100_alv_t
	s_alv_incl	/camelot/oc_alv_incl_s
	t_old_orders	/camelot/oc_gui_0100_alv_t
	data_changed	xfeld
/camelot/oc_gui_0500_s	s_settings	/camelot/oc_sett
/camelot/oc_gui_maint_sel_s	t_matnr_rng	/camelot/oc_matnr_range_t
	t_plant_rng	/camelot/oc_plant_range_t
	t_ordnr_rng	/camelot/oc_ordnr_range_t
	t_created_on_rng	/camelot/oc_dats_range_t
	t_created_by_rng	/camelot/oc_uname_range_t
	t_start_date_rng	/camelot/oc_dats_range_t

	t_end_date_rng	/camelot/oc_dats_range_t
	t_plnum_rng	/camelot/oc_plnum_range_t
	modus	char2
/camelot/oc_matnr_range_s	sign	ddsign
	option	ddoption
	low	matnr
	high	matnr
/camelot/oc_ordnr_range_s	sign	ddsign
	option	ddoption
	low	aufnr
	high	aufnr
/camelot/oc_ord_comment_rfc	aufnr	aufnr
	ord_comment	/camelot/oc_ord_comment
/camelot/oc_plant_range_s	sign	ddsign
	option	ddoption
	low	werks_d
	high	werks_d
/camelot/oc_plnum_range_s	sign	ddsign
	option	ddoption
	low	plnum
	high	plnum
/camelot/oc_uname_range_s	sign	ddsign
	option	ddoption
	low	uname
	high	uname

A.7.3 Datenelemente

Name	Domäne
/camelot/oc_created_by	usnam
/camelot/oc_created_on	datum
/camelot/oc_ord_comment	/camelot/oc_ord_comment

A.7.4 Domänen

Name	Date Type
/camelot/oc_ord_comment	char60

A.8 Screenshots der Anwendung

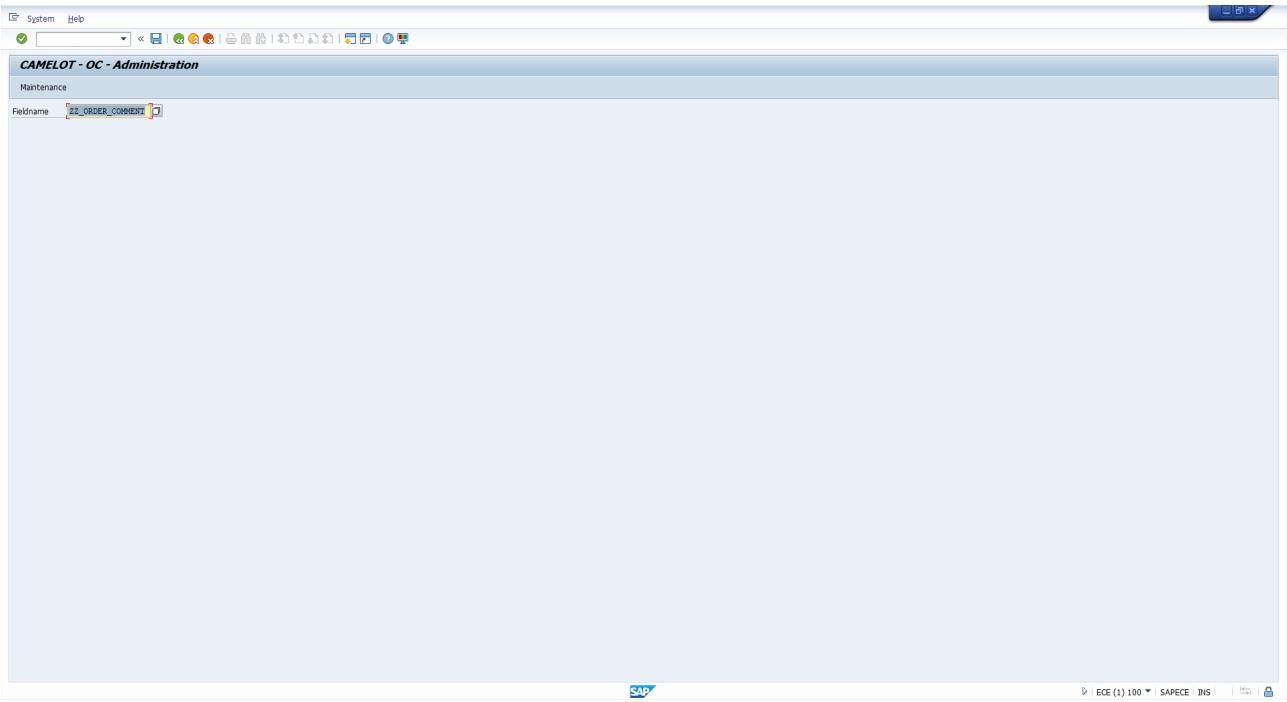


Abbildung 5: Ändern des Feldnamens für den Kommentar in der AUFK Datenbanktabelle

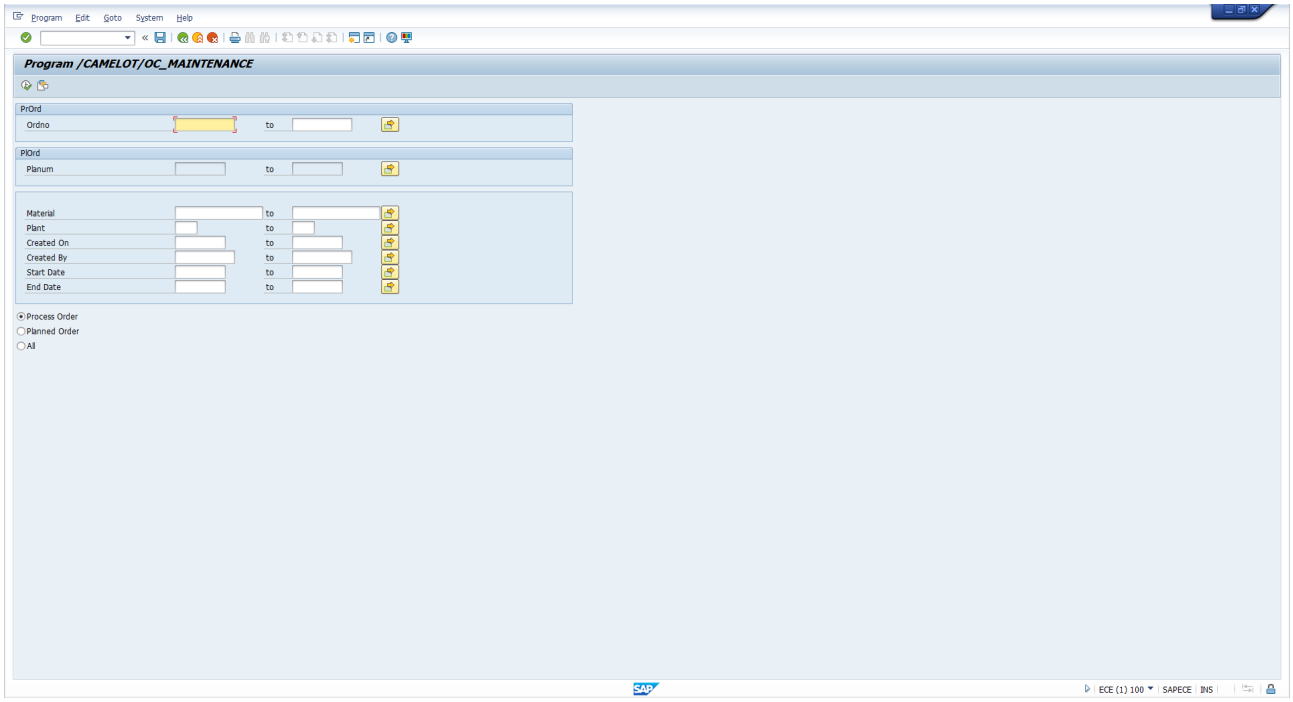
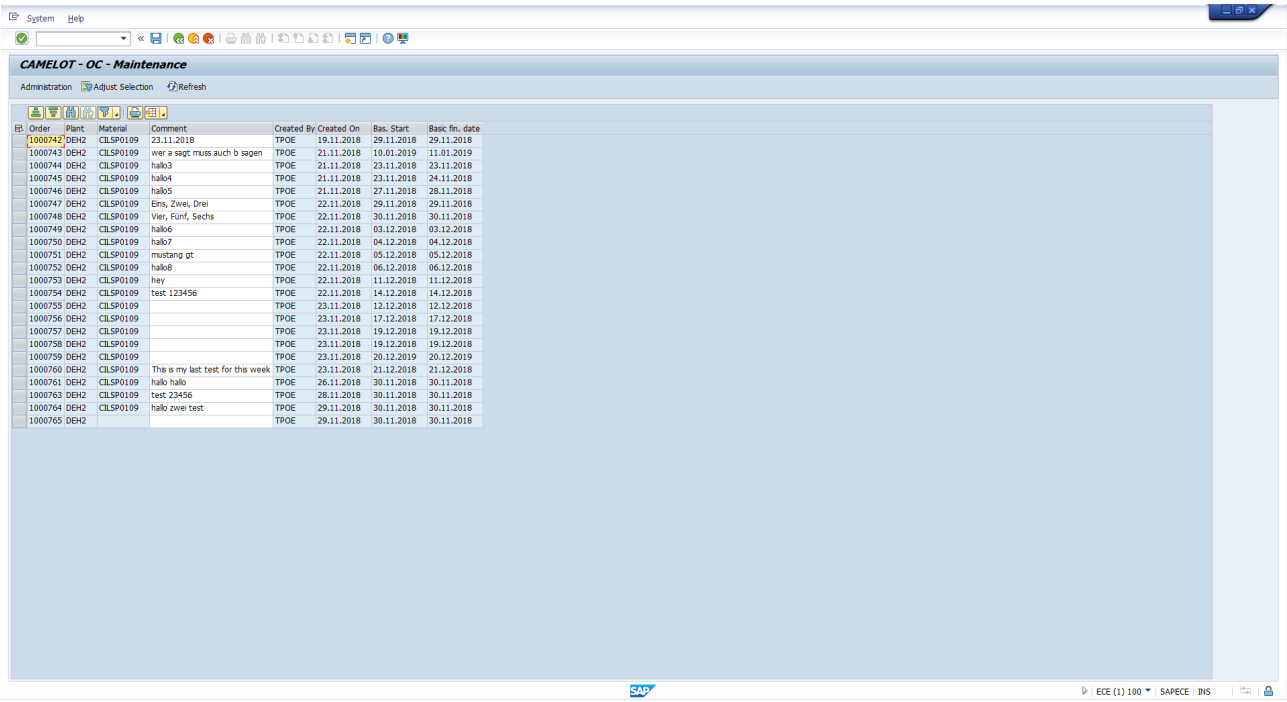
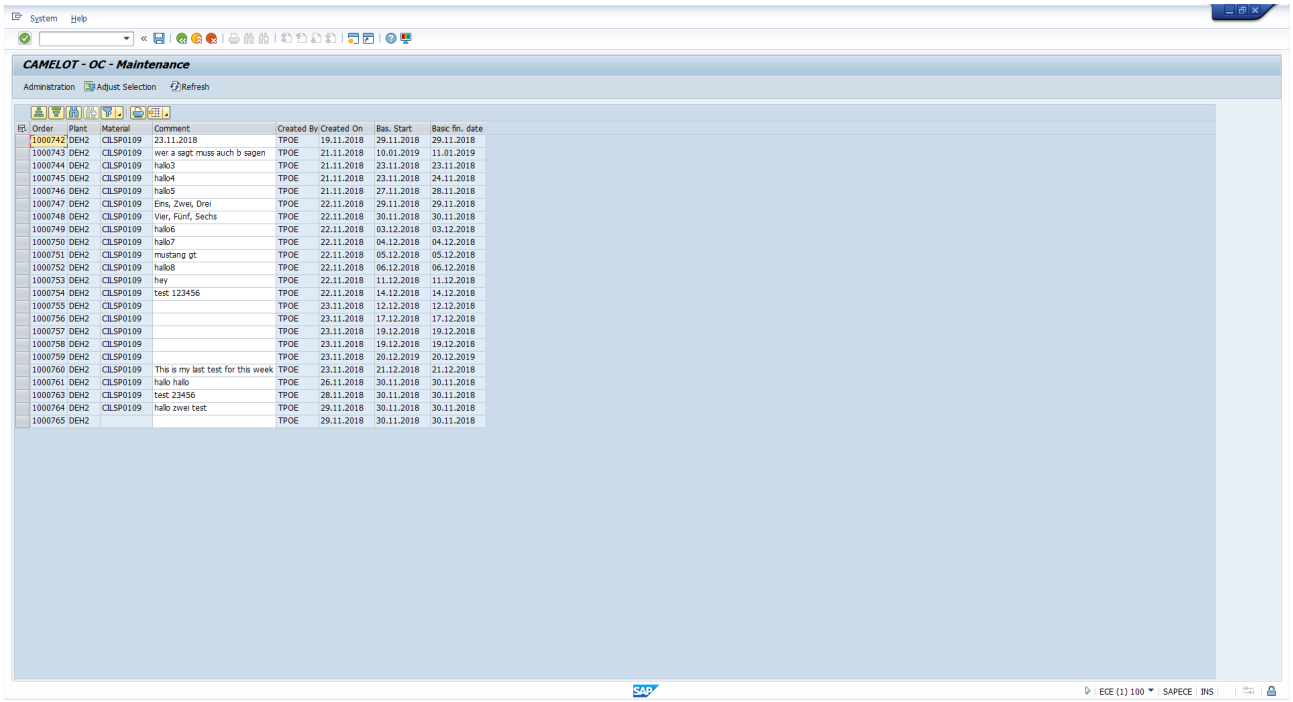


Abbildung 6: Selektionsbildschirm für den Maintenance Screen im ECC



Order	Plant	Material	Comment	Created By	Created On	Bas. Start	Basic fin. date
1000742	DEH2	CILSP0109	23.11.2018	TPOE	19.11.2018	29.11.2018	29.11.2018
1000743	DEH2	CILSP0109	wer a sagt muss auch b sagen	TPOE	21.11.2018	10.01.2019	11.01.2019
1000744	DEH2	CILSP0109	hallo3	TPOE	21.11.2018	23.11.2018	23.11.2018
1000745	DEH2	CILSP0109	hallo4	TPOE	21.11.2018	23.11.2018	24.11.2018
1000746	DEH2	CILSP0109	hallo5	TPOE	21.11.2018	27.11.2018	28.11.2018
1000747	DEH2	CILSP0109	Eins, Zwei, Drei	TPOE	22.11.2018	29.11.2018	29.11.2018
1000748	DEH2	CILSP0109	Vier, Fünf, Sechs	TPOE	22.11.2018	30.11.2018	30.11.2018
1000749	DEH2	CILSP0109	hallo6	TPOE	22.11.2018	03.12.2018	03.12.2018
1000750	DEH2	CILSP0109	hallo7	TPOE	22.11.2018	04.12.2018	04.12.2018
1000751	DEH2	CILSP0109	mustang gt	TPOE	22.11.2018	05.12.2018	05.12.2018
1000752	DEH2	CILSP0109	hallo8	TPOE	22.11.2018	06.12.2018	06.12.2018
1000753	DEH2	CILSP0109	hey	TPOE	22.11.2018	11.12.2018	11.12.2018
1000754	DEH2	CILSP0109	test 123456	TPOE	22.11.2018	14.12.2018	14.12.2018
1000755	DEH2	CILSP0109		TPOE	23.11.2018	12.12.2018	12.12.2018
1000756	DEH2	CILSP0109		TPOE	23.11.2018	17.12.2018	17.12.2018
1000757	DEH2	CILSP0109		TPOE	23.11.2018	19.12.2018	19.12.2018
1000758	DEH2	CILSP0109		TPOE	23.11.2018	19.12.2018	19.12.2018
1000759	DEH2	CILSP0109		TPOE	23.11.2018	20.12.2019	20.12.2019
1000760	DEH2	CILSP0109	This is my last test for this week	TPOE	23.11.2018	21.12.2018	21.12.2018
1000761	DEH2	CILSP0109	hallo hallo	TPOE	26.11.2018	30.11.2018	30.11.2018
1000763	DEH2	CILSP0109	test 23456	TPOE	28.11.2018	30.11.2018	30.11.2018
1000764	DEH2	CILSP0109	hallo zwei test	TPOE	29.11.2018	30.11.2018	30.11.2018
1000765	DEH2			TPOE	29.11.2018	30.11.2018	30.11.2018

Abbildung 7: Aufträge im ALV auf dem Maintenance Screen im ECC



Order	Plant	Material	Comment	Created By	Created On	Bas. Start	Basic fin. date
1000742	DEH2	CILSP0109	23.11.2018	TPOE	19.11.2018	29.11.2018	29.11.2018
1000743	DEH2	CILSP0109	wer a sagt muss auch b sagen	TPOE	21.11.2018	10.01.2019	11.01.2019
1000744	DEH2	CILSP0109	hallo3	TPOE	21.11.2018	23.11.2018	23.11.2018
1000745	DEH2	CILSP0109	hallo4	TPOE	21.11.2018	23.11.2018	24.11.2018
1000746	DEH2	CILSP0109	hallo5	TPOE	21.11.2018	27.11.2018	28.11.2018
1000747	DEH2	CILSP0109	Eins, Zwei, Drei	TPOE	22.11.2018	29.11.2018	29.11.2018
1000748	DEH2	CILSP0109	Vier, Fünf, Sechs	TPOE	22.11.2018	30.11.2018	30.11.2018
1000749	DEH2	CILSP0109	hallo6	TPOE	22.11.2018	03.12.2018	03.12.2018
1000750	DEH2	CILSP0109	hallo7	TPOE	22.11.2018	04.12.2018	04.12.2018
1000751	DEH2	CILSP0109	mustang gt	TPOE	22.11.2018	05.12.2018	05.12.2018
1000752	DEH2	CILSP0109	hallo8	TPOE	22.11.2018	06.12.2018	06.12.2018
1000753	DEH2	CILSP0109	hey	TPOE	22.11.2018	11.12.2018	11.12.2018
1000754	DEH2	CILSP0109	test 123456	TPOE	22.11.2018	14.12.2018	14.12.2018
1000755	DEH2	CILSP0109		TPOE	23.11.2018	12.12.2018	12.12.2018
1000756	DEH2	CILSP0109		TPOE	23.11.2018	17.12.2018	17.12.2018
1000757	DEH2	CILSP0109		TPOE	23.11.2018	19.12.2018	19.12.2018
1000758	DEH2	CILSP0109		TPOE	23.11.2018	19.12.2018	19.12.2018
1000759	DEH2	CILSP0109		TPOE	23.11.2018	20.12.2019	20.12.2019
1000760	DEH2	CILSP0109	This is my last test for this week	TPOE	23.11.2018	21.12.2018	21.12.2018
1000761	DEH2	CILSP0109	hallo hallo	TPOE	26.11.2018	30.11.2018	30.11.2018
1000763	DEH2	CILSP0109	test 23456	TPOE	28.11.2018	30.11.2018	30.11.2018
1000764	DEH2	CILSP0109	hallo zwei test	TPOE	29.11.2018	30.11.2018	30.11.2018
1000765	DEH2			TPOE	29.11.2018	30.11.2018	30.11.2018

Abbildung 8: Selektions Popup auf dem Maintenance Screen

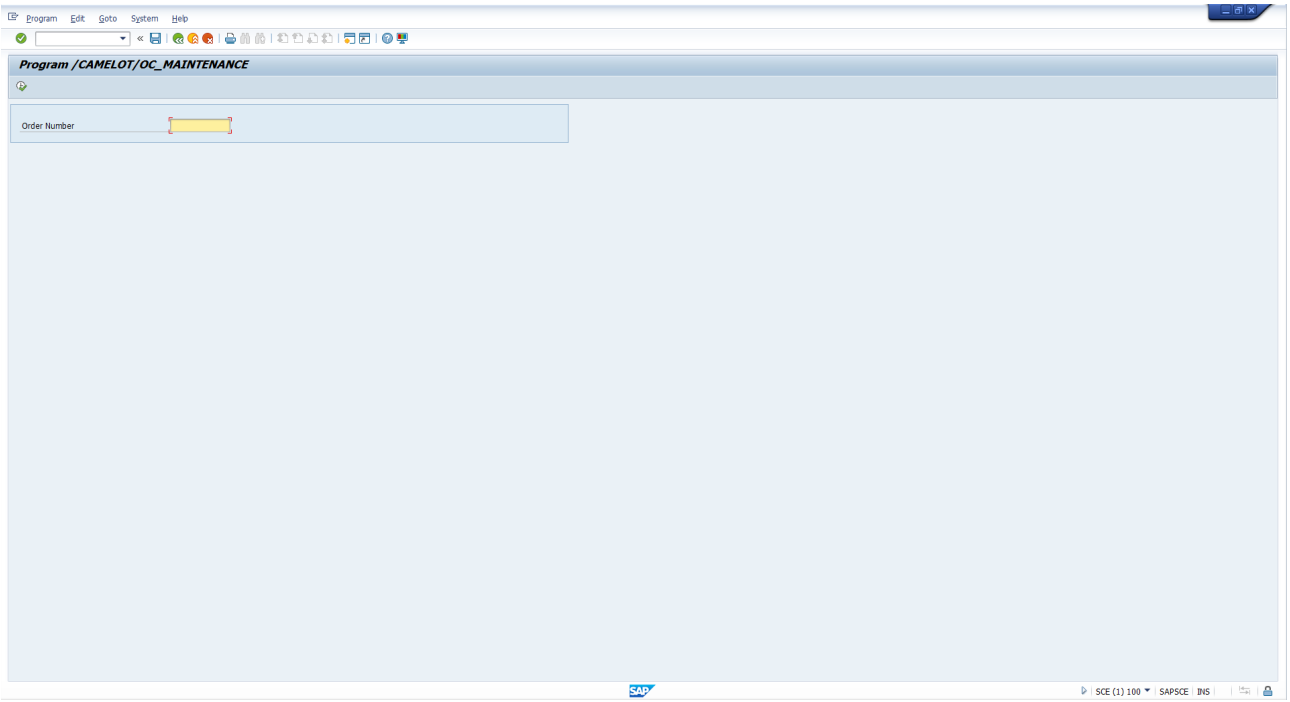


Abbildung 9: Selektionsbildschirm für den Maintenance Screen im APO

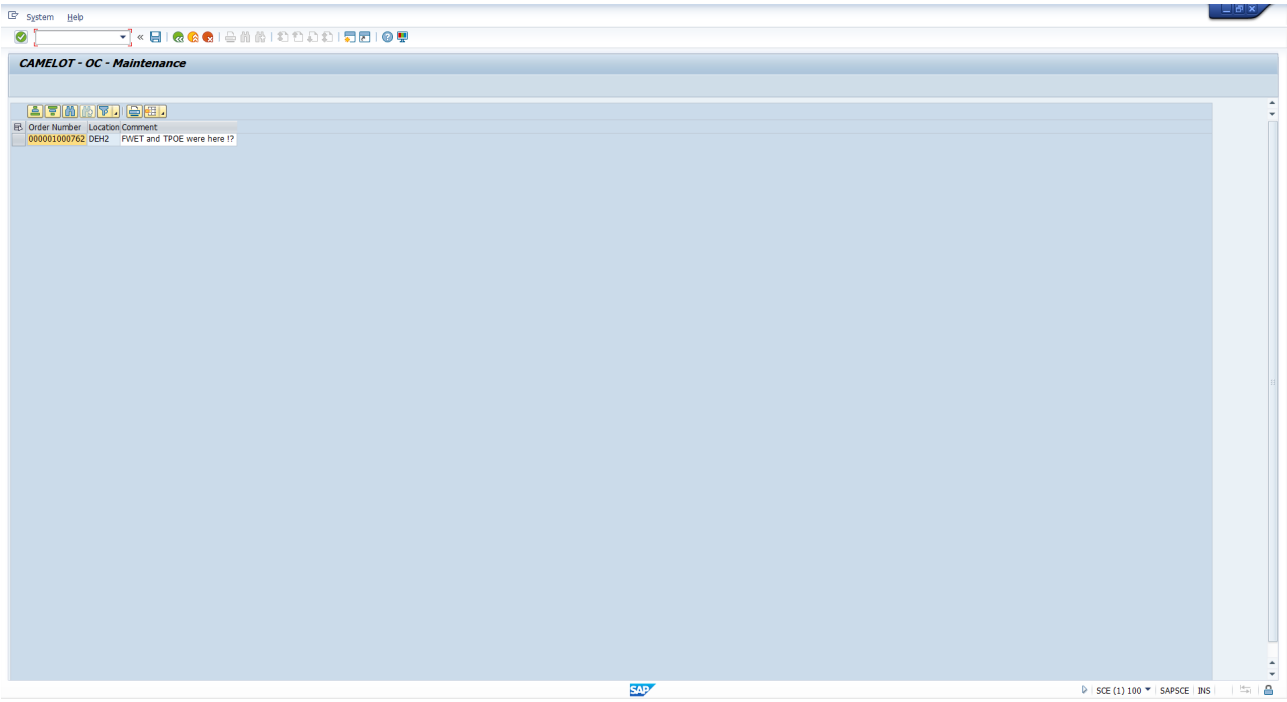


Abbildung 10: Aufträge im ALV auf dem Maintenance Screen im APO

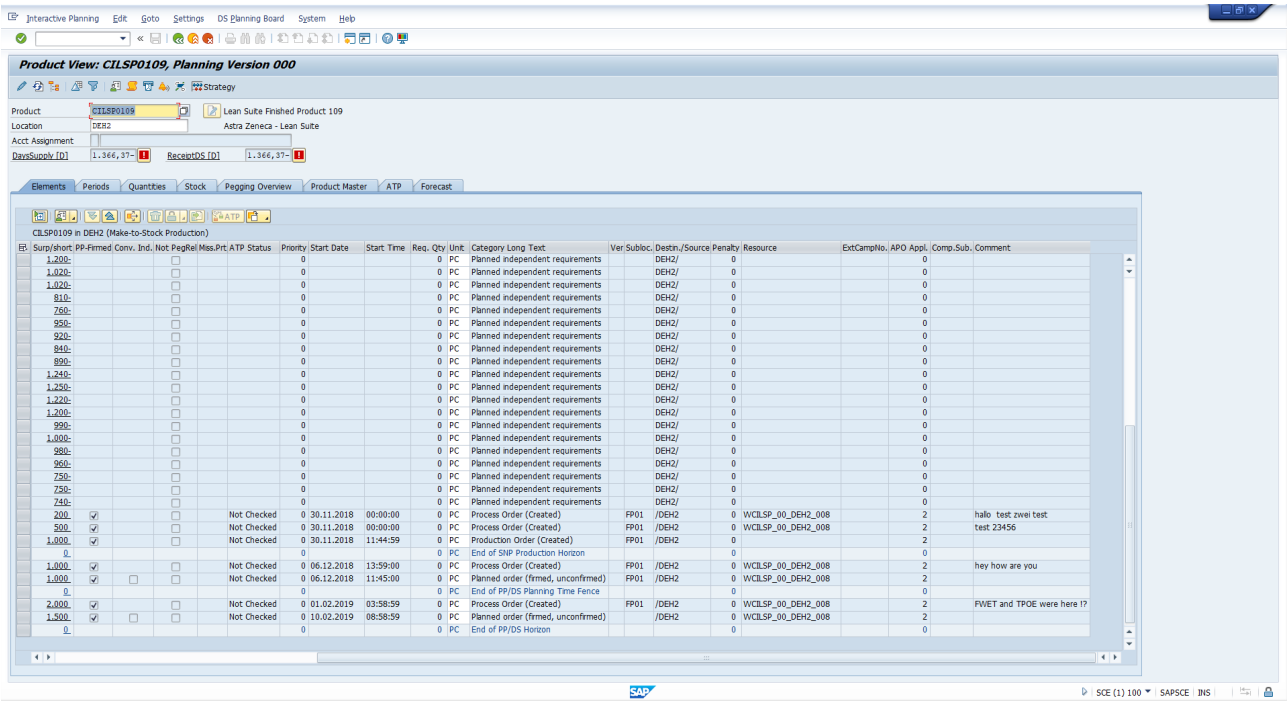


Abbildung 11: Kommentare in der RRP3

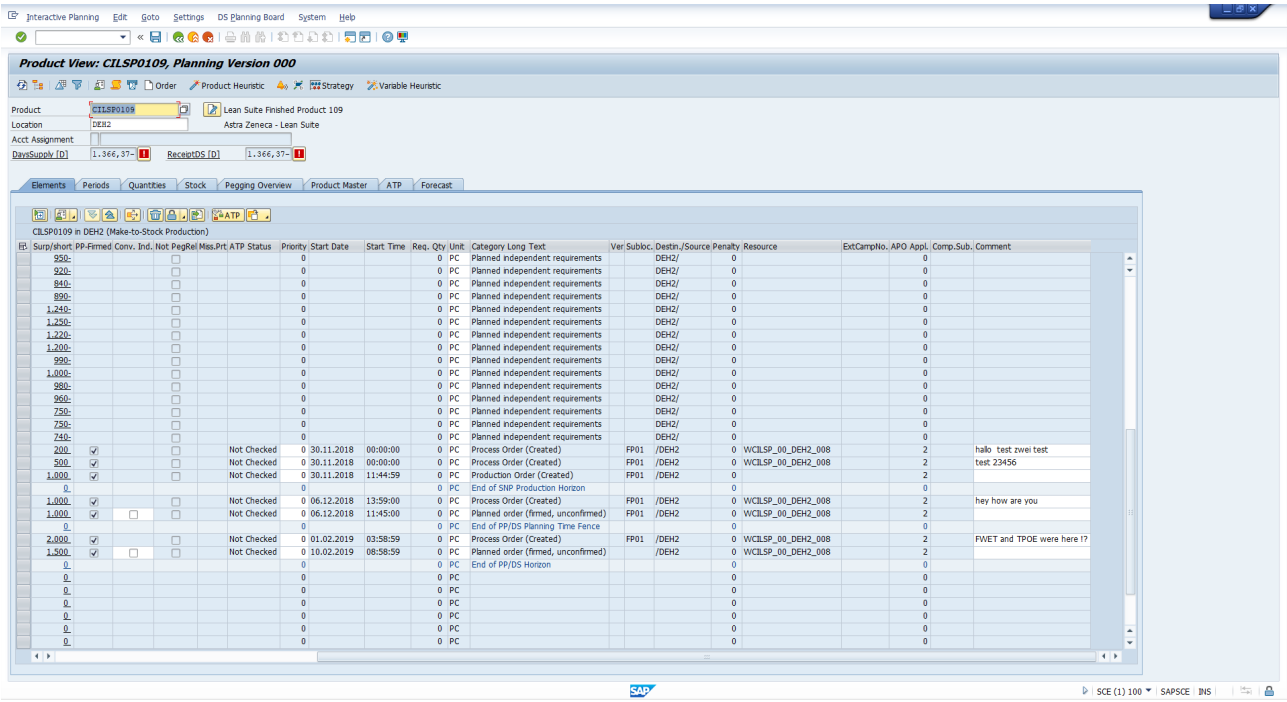


Abbildung 12: Kommentare in der RRP3 zum Editieren

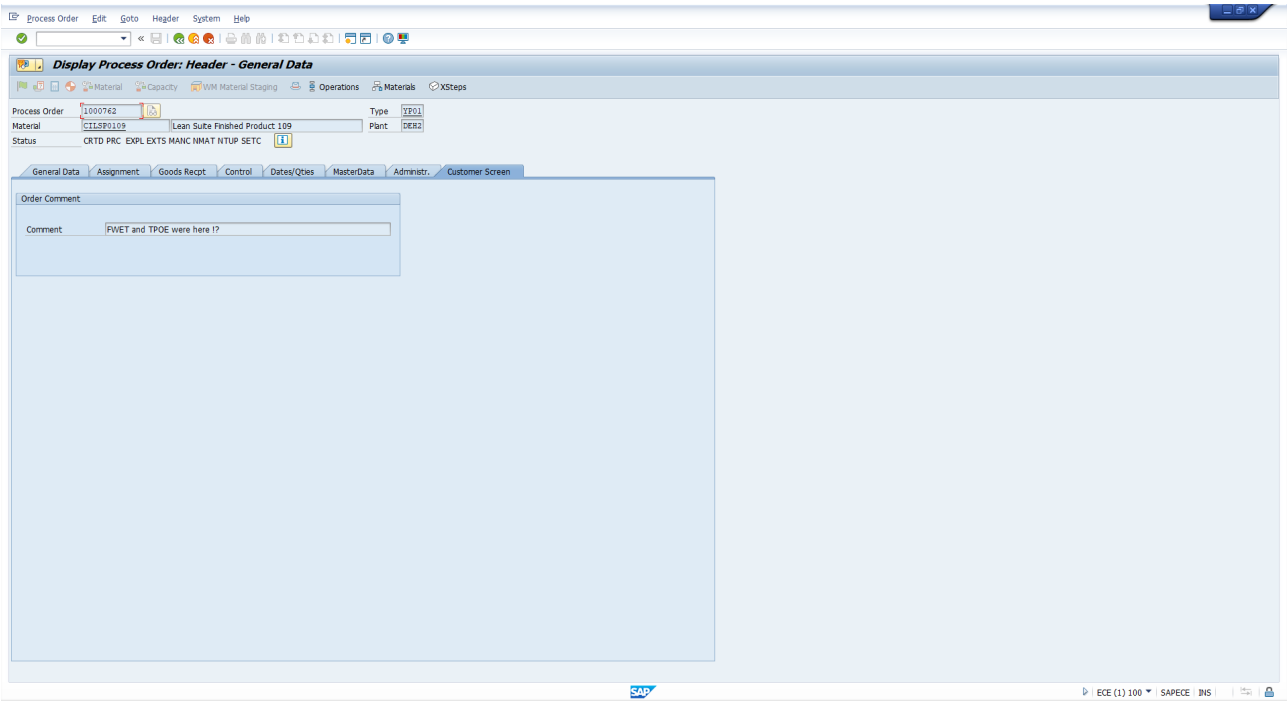


Abbildung 13: Kommentar in der COR3

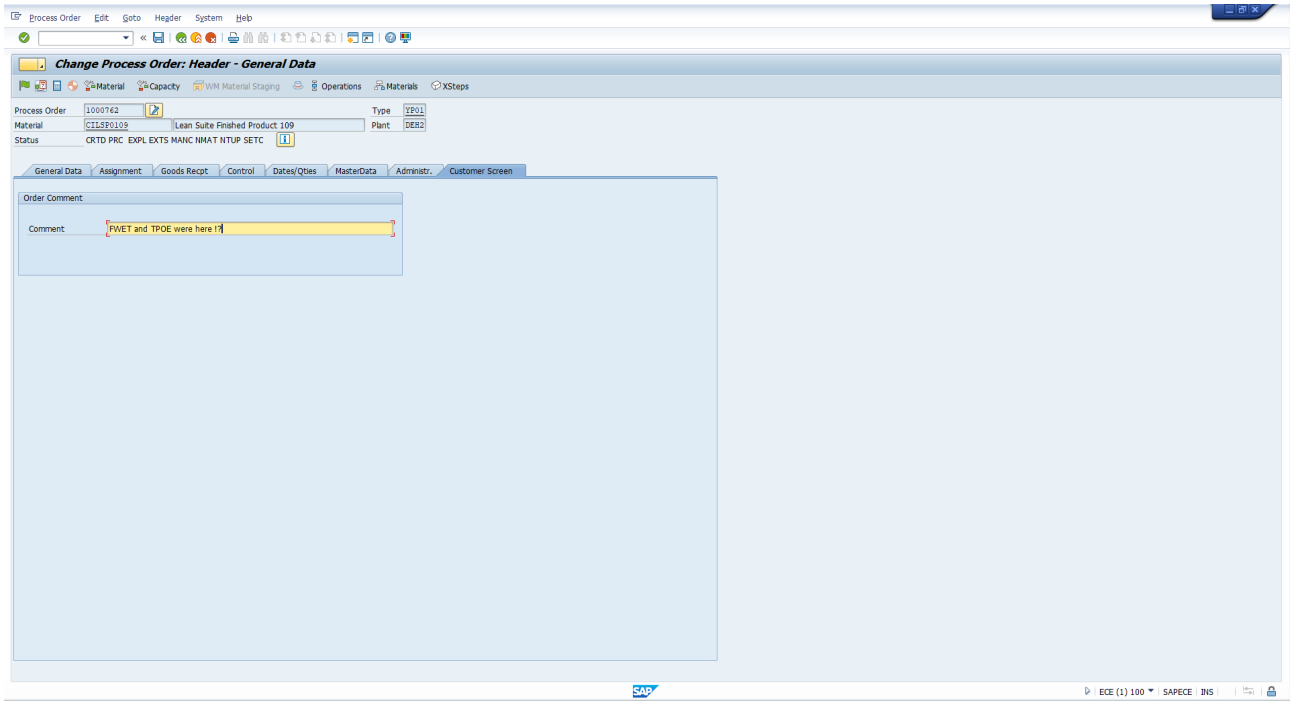


Abbildung 14: Kommentar in der COR2 zum Bearbeiten

A.9 Programmierrichtlinien

Im folgenden Abschnitt werden kurz die am häufigsten gebrauchten Programmierrichtlinien des CAMELOT ITLabs aufgeführt.

A.9.1 Benennung

Variablen Globale Variablen z.B. Variablen in Reports, müssen immer mit einem g(global) anfangen. Lokale Variablen z.B. in Methoden von Klassen, müssen immer mit einem l(local) anfangen.

Darauf folgt dann entweder ein:

1. t für Tabellen
2. s für Strukturen
3. v für Variablen

Darauf folgt dann ein _ und ein passender Name, welcher die Variable möglichst treffsicher beschreibt.

Klassen: Klassen Bezeichnungen starten immer mit dem Paketnamen in welchem sich die Klassen befindet + CL(Class) z.B. /CAMELOT/CL_OC. Der Paketname ist /CAMELOT/OC und in die Mitte wird ein CL gehängt.

A.9.2 Formatierung

Pretty-Printer: Der Pretty Printer, ein Tool welches den Quellcode aufbereitet und somit dem Entwickler eine bessere Lesbarkeit bietet muss ,wie in der folgenden Grafik dargestellt, eingestellt werden. Der Pretty-Printer ist über die Schaltfläche [Pretty Printer] oder die Tastenkombination Ümschalt + F1äufzurufen.

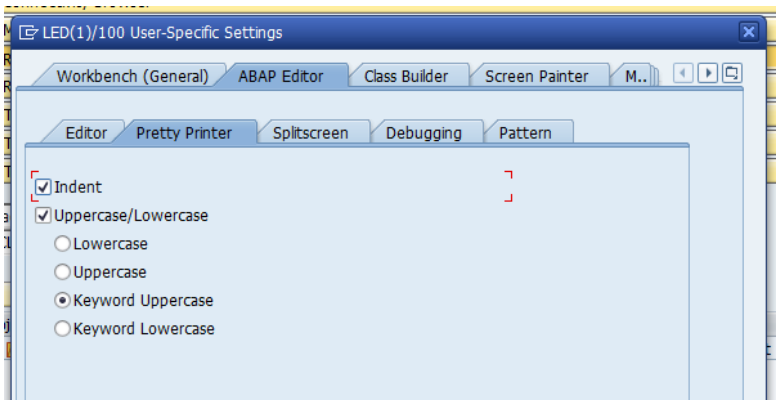


Abbildung 15: Einstellungen des Pretty-Printers

A.10 Klassendiagramm

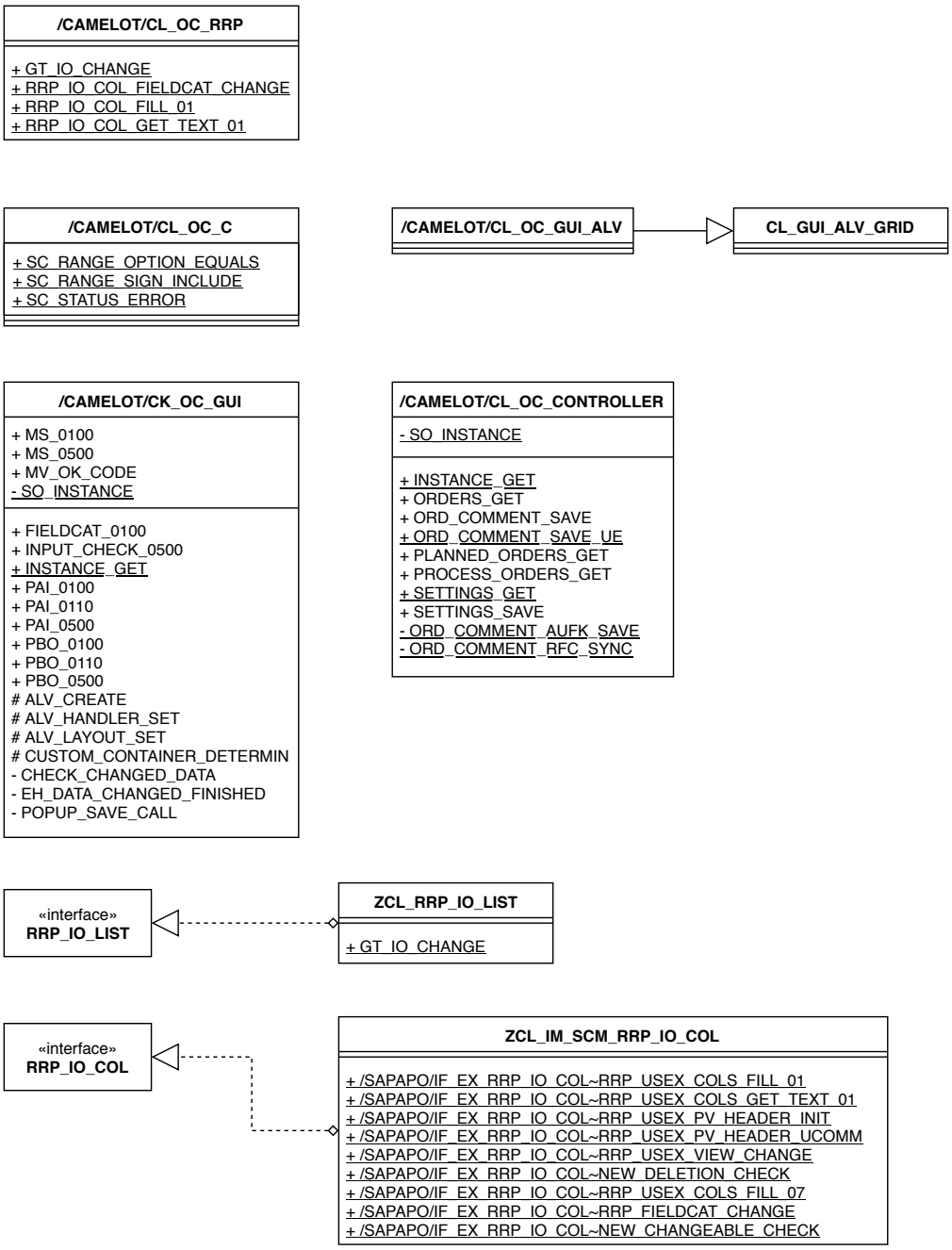


Abbildung 16: Klassendiagramm