

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 Planer

Abgabetermin: Mannheim, den 15.12.2018

Prüfungsbewerber:

Thomas Pöhlmann
Schwingstraße 10
68199 Mannheim



Ausbildungsbetrieb:

CAMELOT ITLAB
Theodor-Heuss-Anlage. 12
68165 Mannheim

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

Inhaltsverzeichnis

Abbildungsverzeichnis	III
Tabellenverzeichnis	IV
Listings	V
Abkürzungsverzeichnis	VI
Transaktionsverzeichnis	VII
1 Einleitung	1
1.1 Vorstellung des Betriebs und meiner Selbst	1
1.2 Projektziel	1
1.3 Projektumfeld	1
1.4 Projektbeteiligte Personen	1
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 „Make or Buy“-Entscheidung	4
3.2.2 Projektkosten	4
3.2.3 Amortisationsdauer	5
3.3 Nutzwertanalyse	7
3.4 Anwendungsfälle	7
3.5 Qualitätsanforderungen	7
3.6 Lastenheft/Fachkonzept	7
3.7 Zwischenstand	7
4 Entwurfsphase	8
4.1 Zielplattform	8
4.2 Architekturdesign	8
4.3 Entwurf des Userinterface	8
4.4 Datenmodell	9
4.5 Maßnahmen zur Qualitätssicherung	10
4.6 Pflichtenheft/Datenverarbeitungskonzept	10

Inhaltsverzeichnis

4.7	Zwischenstand	10
5	Implementierungsphase	11
5.1	Iterationsplanung	11
5.2	Implementierung der Datenstruktur ECC	11
5.3	Implementierung der Benutzeroberfläche ECC	11
5.4	Implementierung PBO und PAI	12
5.5	Implementierung der Geschäftslogik ECC	12
5.6	Implementierung der COR Erweiterung	12
5.7	Implementierung der Datenstruktur APO	13
5.8	Implementierung der Benutzeroberfläche APO	13
5.9	Implementierung der Geschäftslogik APO	13
5.10	Implementierung der RRP3 Erweiterung	14
5.11	Zwischenstand	14
5.12	Code Reviews während des Projekts	15
5.13	Manuelle Tests	15
5.14	Bug Fixing	15
6	Fazit	15
6.1	Soll-/Ist-Vergleich	15
6.2	Ausblick	15
	Literaturverzeichnis	16
	Eidesstattliche Erklärung	17
A	Anhang	i
A.1	Detaillierte Zeitplanung	i
A.2	Lastenheft (Auszug)	ii
A.3	Iterationsplan	iii
A.4	Use Case-Diagramm	iv
A.5	Pflichtenheft (Auszug)	iv
A.6	Dictionary-Objekte	vii
A.6.1	Tabellentypen	vii
A.6.2	Strukturen	vii
A.6.3	Datenelemente	viii
A.6.4	Domänen	viii
A.7	Screenshots der Anwendung	ix
A.8	Programming Guidlines	xix
A.9	Klassendiagramm	xx

Abbildungsverzeichnis

1	Vereinfachtes ER-Modell	9
2	Vereinfachtes ER-Modell	9
3	Vereinfachtes ER-Modell	10
4	Use Case-Diagramm	iv
5	Ändern des Feldnamens für den Kommentar in der AUFK Datenbanktabelle	ix
6	Selektionsbildschirm für den Maintenance Screen	x
7	xi
8	xii
9	xiii
10	xiv
11	xv
12	xvi
13	xvii
14	xviii
15	Klassendiagramm	xx

Tabellenverzeichnis

1	BeteiligtePersonen	2
2	Zeitplanung	3
3	HardwareKosten	5
4	ProjektKosten	6
5	Zwischenstand nach der Analysephase	8
6	Zwischenstand nach der Entwurfsphase	10
7	Zwischenstand nach der Implementierungsphase	14

Listings

Abkürzungsverzeichnis

ALV	ABAP List View
APO	Advanced Planning and Optimization
ECC	ERP Central Component
PBO	Process Before Output
PAI	Process After Input
UML	Unified Modeling language

Transaktionsverzeichnis

COR1-3:

Diese Transaktion ist zum Erstellen, bearbeiten und anzeigen von Prozess Aufträgen

RRP3:

1 Einleitung

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

1.1 Vorstellung des Betriebs und meiner Selbst

Ich habe meine Ausbildung am 01.März 2017 bei der Spreitzenbarth Consultants GmbH angefangen. Einem kleinen Unternehmen mit 50 Mitarbeitern mit Fokus auf Supply Chain Management, dort habe ich hauptsächlich Web Anwendungen mit ASP.NET und AngularJs gemacht. Aufgrund einer Firmen Auflösung habe ich zum 01.06.2018 meinen Ausbildungsbetrieb gewechselt und bin zu Camelot gekommen. Dort habe ich mich dann mit ABAP und dem SAP Umfeld betätigt.

1.2 Projektziel

Im SAP APO bzw. im SAP ECC hat der Planer die Möglichkeit Prozess- oder Plan-aufträge manuell anzulegen. Allerdings gibt es hier keine Möglichkeit eine Notiz oder eine Beschreibung zu schreiben. Das Projektziel ist die Erstellung von Programmen, die dem Planer ermöglichen, Kommentare für Aufträge zu erstellen und anzusehen. Außerdem sollen bereits vorhandene Programm erweitert werden, um dem Planer den bestmöglichen Komfort zu bieten. Da dieses Projekt ein internes Projekt ist, wurden alle Anforderungen in innerbetrieblichen Besprechungen ausgemacht. Bei der Implementierung wird darauf geachtet, dass später ein Kommentarverlauf hinzugefügt wird, 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 wird, ist für die softwareseitige Implementierung technischer Anforderungen zuständig. Der Fokus der Abteilung liegt auf Supply Chain Management, im Speziellen Produktions- und Feinplanung.

1.4 Projektbeteiligte Personen

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

1 Einleitung

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

Tabelle 1: Beteiligte Personen

1.5 Projektbegründung

2 Projektplanung

2.1 Projektphasen

Zur Umsetzung des Projekts standen mir 70 Stunden zur Verfügung. Diese waren sowohl für die Umsetzung des Projekts als auch für die Dokumentation. Vor Projektbeginn wurde das Projekt in mehrere Phasen gesplittet und die Stunden aufgeteilt. Eine grobe Übersicht kann der nachfolgenden Tabelle entnommen werden.

Tabelle 2 zeigt ein Beispiel für eine grobe Zeitplanung.

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, welche sich im Anhang befindet, sind alle Ressourcen aufgelistet, die für das Projekt eingesetzt wurden. Damit sind sowohl Hardware, Software und das Personal gemeint. Es wurde darauf geachtet, dass nur Software zum Einsatz kommt welche entweder kostenfrei (z.B. als Freeware oder gar Open Source) angeboten werden oder die Camelot AG bereits Lizenzen für diese besitzt und zur Verfügung stellen kann. Dadurch sollen die Projektkosten möglichst geringgehalten werden.

2.3 Entwicklungsprozess

TODO

3 Analysephase

3.1 Ist-Analyse

Das Projekt wird im Umfeld der Produktionsplanung durchgeführt und bezieht sich auf alle Zutun Abgangselemente. Der Planer hat im SAP Advanced Planning and Optimization (APO) die Möglichkeit Zugangs und Abgangselemente manuell anzulegen. In Folgeprozessen kann es zu Problemen führen, wenn die Gründe dieser Planänderungen nicht sichtbar gemacht werden. Bisher müssen solche Änderungen ohne Systemunterstützung per Email oder auf anderem Wege an alle beteiligten mitgeteilt werden.

3.2 Wirtschaftlichkeitsanalyse

Aufgrund der Probleme, die bereits in der Projektbegründung und der Ist-Analyse beschrieben wurden, ist dieses Projekt absolut notwendig. Trotzdem werde ich in dem folgenden Abschnitt analysieren ob die Umsetzung auch aus Wirtschaftlichen Gesichtspunkten gerechtfertigt ist.

3.2.1 „Make or Buy“-Entscheidung

Es gibt nichts vergleichbares auf dem Markt was in dieser Art und weise angeboten wird.

Wahl zwischen machen und Entwickler kaufen und programmieren lassen

Irwelche Inder kosten * Zeit. Später support schwierig da man entweder die Entwickler halten muss oder sie halt weggehen lässt.

Unkalkulierbare kosten nach Projektabschluss. Da wir davon leben Produkte zu entwickeln und zu lizensieren.

3.2.2 Projektkosten

Die Projektkosten, die während dem gesamten Entwicklungszeitraum anfangen sollen im Folgenden kalkuliert werden. Dafür müssen neben den Kosten, die für Hardware und Software anfallen auch die Personalkosten berücksichtigt werden. Da die genauen Personalkosten Betriebsgeheimnis sind, wird die Kalkulation mit groben Stundensätzen durchgeführt. Der Stundensatz eines Auszubildenden ergibt sich aus dem Monats ca. 1000€.

3 Analysephase

$$365 \text{ Tage/Jahr} \cdot \frac{5}{7} - 30 \text{ Tage} = 260 \text{ Tage/Jahr} \quad (1)$$

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

$$1000 \text{ €/Monat} \cdot 12 \text{ Monate/Jahr} = 12000 \text{ €/Jahr} \quad (3)$$

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

Der Rechnung zufolge beträgt der Stundensatz eines Azubis ca. 7€ die Stunde und die Kosten eines normalen Mitarbeiters schätze ich auf 25€ die Stunde. Für die Ressourcen werden die gesamten Hardwarekosten addiert und dann durch die durchschnittliche Lebenszeit in Arbeitsstunden, d.h. $3 \cdot 365 \cdot 5/7 \cdot 8$ 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

Daraus ergibt sich dann folgende Rechnung und Stundensatz

$$\frac{1680 \text{ €/3 Jahre}}{3} = 560/\text{Jahr} \quad (5)$$

$$\frac{560 \text{ €/Jahre}}{1840 \text{ h/Jahr}} \approx 0.30 \text{ €/h} \quad (6)$$

Es ergibt sich also ein Stundensatz von 0.3€ für die Hardware Kosten. Dazu kommen natürlich noch Kosten für Arbeitsplatz, Möblierung, Strom und Internet.

3.2.3 Amortisationsdauer

Im folgenden Abschnitt soll berechnet werden, ab welchem Zeitraum sich die Entwicklung auch aus wirtschaftlichem Sichtpunkt für das Camelot ITLab und für den Kunden lohnt. Die Amortisationsdauer wird berechnet indem man die Produktionskosten bzw. Anschaffungskosten durch die Gewinne dividiert, welche durch das neue Produkt erzielt wurden.

3 Analysephase

Vorgang	Mitarbeiter	Auszubildender	Zeit	Personal	Ressourcen	Gesamt
Anforderungs- aufnahme	2	1	2h	64	60	124
Ressourcen Ent- wicklung		1	39h	273	390	663
Testphase		1	3h	21	30	51
Code Reviews	2	1	3h	171	90	281
Fazit und Doku- mentation		1	23h	161	230	391
Gesamt						1510€

Tabelle 4: ProjektKosten

Camelot ITLab Die Gewinne welche durch dieses Produkt entstehen lassen sich in zwei Kategorien teilen. Zum einen die Gewinne welche durch den tatsächlichen verkauf dieses Produkt erzielt werden und zum anderen die Gewinne die erzielt werden, wenn der Kunde außerdem neben diesem Tool dann auch noch andere Software der Camelot kaufen möchte. Für diese Tool wird ein Preis von etwa 500€ angesetzt. Der Break-even-point (Gewinnschwell) liegt bei

$$\frac{1510 \text{ €}}{500 \text{ €}} \approx 3 \quad (7)$$

Kunde In Fall dieses Projektes lassen sich die Gewinne nicht so leicht erfassen, da zum einen die Fehleranfälligkeit der alten Lösung, Kommentare per Email verschicken, Zettelwirtschaft, usw. deutlich verringert wird. Zum Anderen erspart das Tool dem Planer, welcher die Prozess anlegt deutlich Zeit, da dieser nicht noch ein weiteres Programm benötigt um die Kommentare einzutragen und an seine Kollegen zu verteilen und den Mitarbeitern an der Maschine Zeit, welche die Kommentare direkt in der Transaktion neben den Aufträgen sehen deutlich Zeit.

Beispielrechnung (verkürzt) Es wird davon ausgegangen, dass ein Planer jeden Tag ca. eine Stunde damit beschäftigt ist Aufträge zu Kommentieren. Dieses Tool bietet eine Zeitersparung von ca. 50% da die Texte weiterhin per Hand verfasst werden müssen. Dies bedeutet Zeiteinsparung von 30 Minuten pro Tag und pro Planer. Bei etwa 230 Arbeitstagen pro Jahr ergibt sich eine gesamte Zeiteinsparung von

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

Das Gehalt eines Produktionsplaners beträgt laut Experten etwa 14€ die Stunde. Die Tatsächlichen kosten, die für die Firma anfallen, werden auf ca. 20€ geschätzt. Dadurch ergibt sich eine jährliche Einsparung von

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

3 Analysephase

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: $\frac{500 \text{ €}}{2 \cdot 2300 \text{ € €/Jahr}} \approx 0,19 \text{ Jahre} \approx \text{Wochen.}$

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

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

3.3 Nutzwertanalyse

Lohnt sich das Projekt für den Kunden was wir gespart weniger fehler usw. alles außer geld

3.4 Anwendungsfälle

An der Maschine kann der arbeiter direkt neben dem auftrag auch den Kommentar sehen.

Use-Case Diagram

3.5 Qualitätsanforderungen

Wie in jedem anderen Projekt des Camelot ITLabs gelten auch hier die programming Guidelines. Ein kleine Zusammenfassung mit allen für dieses Projekt wichtigen Punkten findet sich im Anhang Anhang [A.8: Programming Guidlines](#) auf Seite [xix](#). Außerdem muss es für den Administration Screen einen Eingabeüberprüfung geben, der verhindert, dass der User Keyfelder bzw. Fleder die es nicht gibt angibt.

3.6 Lastenheft/Fachkonzept

- Auszüge aus dem Lastenheft/Fachkonzept, wenn es im Rahmen des Projekts erstellt wurde.
- Mögliche Inhalte: Funktionen des Programms (Muss/Soll/Wunsch), User Stories, Benutzerrollen

Beispiel Ein Beispiel für ein Lastenheft findet sich im Anhang [A.2: Lastenheft \(Auszug\)](#) auf Seite [ii](#).

3.7 Zwischenstand

Tabelle [5](#) zeigt den Zwischenstand nach der Analysephase.

Vorgang	Geplant	Tatsächlich	Differenz
1. Analyse des Ist-Zustands	0.5 h	0.5 h	
2. „Make or buy“-Entscheidung und Wirtschaftlichkeitsanalyse	1 h	1 h	
3. Projektkosten	1 h	1.5 h	
4. Erstellen eines „Use-Case“-Diagramms	2 h	2 h	
5. Erstellen des Lastenhefts	3 h	3 h	

Tabelle 5: Zwischenstand nach der Analysephase

4 Entwurfsphase

4.1 Zielplattform

Das Abschlussprojekt soll wie bereits im Projektziel beschrieben eine Erweiterung zu bereits vorhandenen Transaktionen darstellen als auch eigene Programme zur Verwaltung und Massenflege der Kommentare mit sich bringen. Als Programmiersprache wird ABAP (Advanced Business Application Programming) verwendet, eine eigens von der SAP entwickelte Programmiersprache, die in ihrer Grundstruktur der Sprache COBOL ähnelt.

4.2 Architekturdesign

Die Programme im APO und ERP Central Component (ECC) werden 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 werden in einer Funktionsgruppe definiert und mithilfe des SAP Screen Painter gestaltet. Außerdem gibt es jeweils eine Controller Klasse, welcher 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 startet, da aus einem Report direkt kein Screen aufgerufen werden kann. Das Process Before Output (PBO) und das Process After Input (PAI) in der Funktionsgruppe sind dynamisch agierende Module, welche dann auf die jeweilige Methode der Graphical User Interface (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

4 Entwurfsphase

Möglichen Selektionskriterien vorab geplant damit später keine Zeit mit der Erstellung von unnötigen Datenstrukturen verbraucht wird. Es wird später im ERP Central Component (**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 wird darauf geachtet, dass der code leicht zu erweitern ist, da hier 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 Auftrags Daten und den Kommentaren in einem ALV (ABAP List View) dargestellt. Im APO wird es denselben Hauptscreen auch geben hier fällt allerdings der Administration Screen weg, da hier alle Kommentare unabhängig der Kategorie in der Datenbank /SAPAPO/ORD-FLDS gespeichert werden.

4.4 Datenmodell

Im ECC werden zwei Datenbank Tabellen erstellt. Zum einen die /CAMELOT/OC_COMT. Die Beziehungen sind in dem Folgenden Unified Modeling language (**UML**).

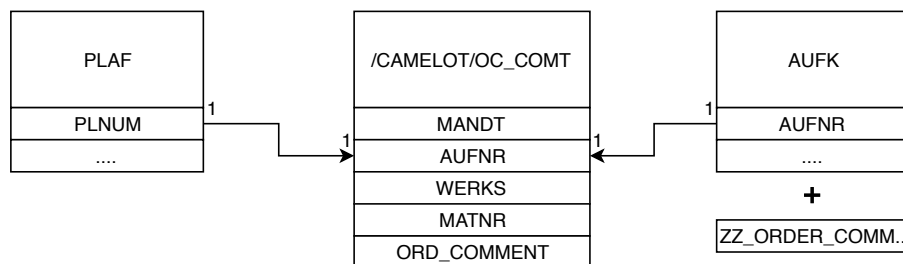


Abbildung 1: Vereinfachtes ER-Modell

Die Tabelle AUFK wird mittels einem Custom Include um ein Feld ZZ_ORDER_COMMENT erweitert. Dieses wird für das COR1 Enhancement gebraucht. Außerdem wird eine Datenbank Tabelle /CAMELOT/OC_SETT erstellt, wie in dem folgenden Unified Modeling language (**UML**) beschrieben. In dieser Datenbank wird vorerst lediglich ein Feldname der AUFK gespeichert. Später muss diese Tabelle dann noch um Tabellennamen erweitert werden.

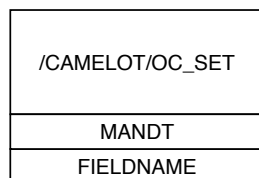


Abbildung 2: Vereinfachtes ER-Modell

Im APO benötigen wir keine Extra Datenbank zum Speichern der Kommentare, da wir hier, wie bereits oben erwähnt, die /SAPAPO/ORDFLDS per Custom Include um ein FELD ORDER_COMMENT erweitern können.

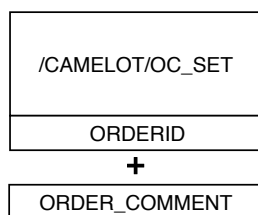


Abbildung 3: Vereinfachtes ER-Modell

4.5 Maßnahmen zur Qualitätssicherung

Um die hohen Qualitätsanforderungen der Camelot ITLab zu gewährleisten und die Qualitätsanforderungen des Projekts zu sichern, werden während der laufenden Entwicklung nach jedem Iterationsschritt die neu eingebauten Funktionalitäten getestet. Außerdem wird es mehrere Code Review geben in denen andere Entwickler sich den Code anschauen und gegebenenfalls Schwachstellen erkennen und Verbesserungsvorschläge einbringen. Alle Tests werden manuell durchgeführt.

4.6 Pflichtenheft/Datenverarbeitungskonzept

- Auszüge aus dem Pflichtenheft/Datenverarbeitungskonzept, wenn es im Rahmen des Projekts erstellt wurde.

Beispiel Ein Beispiel für das auf dem Lastenheft (siehe Kapitel 3.6: [Lastenheft/Fachkonzept](#)) aufbauende Pflichtenheft ist im Anhang A.5: [Pflichtenheft \(Auszug\)](#) auf Seite iv zu finden.

4.7 Zwischenstand

Tabelle 6 zeigt den Zwischenstand nach der Entwurfsphase.

Vorgang	Geplant	Tatsächlich	Differenz
1. Prozessentwurf	2 h	3 h	+1 h
2. Datenbankentwurf	3 h	5 h	+2 h
3. Erstellen von Datenverarbeitungskonzepten	4 h	4 h	
4. Benutzeroberflächen entwerfen und abstimmen	2 h	1 h	-1 h
5. Erstellen eines UML-Komponentendiagramms	4 h	2 h	-2 h
6. Erstellen des Pflichtenhefts	4 h	4 h	

Tabelle 6: Zwischenstand nach der Entwurfsphase

5 Implementierungsphase

5.1 Iterationsplanung

Bevor mit der eigentlichen Implementierung begonnen wurde, wurde zuerst ein Iterationsplan 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.3: Iterationsplan](#) auf Seite [iii](#).

5.2 Implementierung der Datenstruktur ECC

Zuerst wurden alle Dictionary-Objekte, welche im ERP Central Component ([ECC](#)) gebraucht werden, erstellt. Eine Vollständige Liste aller Tabellen Typen, Strukturen, Datenelemente und Domänen können dem Anhang entnommen werden. Außerdem wurde die Datenbank Tabellen wie unter 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 Graphical User Interface ([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 ABAP List View ([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 Process Before Output ([PBO](#)) und das Process After Input ([PAI](#)) der Funktionsgruppe wurde dynamisch programmiert. Das bedeutet, dass je nach Screen die zugehörige Methode in der Graphical User Interface ([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. Screenshots der Anwendung befinden sich unter [Anhang A.7: Screenshots der Anwendung](#) auf Seite [ix](#).

Beispiel Screenshots der Anwendung in der Entwicklungsphase mit Dummy-Daten befinden sich im [Anhang A.7: 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 wir das **PBO** betreten, sondern nur, wenn die ALV Struktur, welche in der Member Struktur des jeweiligen Screens ist, Initial also "leer" ist. Dadurch wird verhindert dass wir zu viele Datenbank Zugriffe haben, da jedes Mal die Daten neu geladen werden. Nichts desto trotz können wir einen Refresh einleiten indem wir einfach die ABAP List View (**ALV**) Struktur Clearen. Im **PAI** werden all User Befehle (OK Code) abgefangen welche vom Screen geworfen werden und dann die jeweilige Action ausgeführt.

5.5 Implementierung der Geschäftslogik ECC

Die eigentliche Geschäftslogik und Datenbank zugriffe 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 drin 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, zum einen einen Screen Exit und zum anderen zwei Function Exits vor und nach dem laden der Daten hat. Verwendet wurde das Enhancement PPCO0020. Diese bietet alle Komponenten, die ich zum anzeigen der Daten benötige. Zuerst wurde das Screen Exit mit einem Label und einem Eingabefeld erweitert und aktiviert. Dann wurde das **PBO** Modul um eine Methode erweitert, dass das Eingabefeld in der COR3 deaktiviert wird, da diese Transaktion nur zum Anzeigen ist. Außerdem wurde in dem Top Include eine globale Struktur 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 Kommentar von Prozessaufträgen zu ändern bzw. beim anlegen eines Auftrags anzugeben wird noch ein weiterer Funktionsbaustein benötigt, da der oben genannte, keinen Function Exit für das Speichern hat, von wo wir unsere eigene Speicherlogik aus aufrufen können. 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 wo zu außerdem 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, muss nun ein großteil der selben Datenstrukturen auch im APO angelegt werden. Eine Liste aller Dictionary-Objekte kann dem Anhang entnommen werden. Außerdem 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.

Beispiel Screenshots der Anwendung in der Entwicklungsphase mit Dummy-Daten befinden sich im Anhang A.7: 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, welchem 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 Order Number filtern kann.

5.9 Implementierung der Geschäftslogik APO

Wie im ECC findet auch im APO die gesamte Geschäftslogik im Controller statt. Neben den selben Klassen welche es auch im ECC gibt(Constants, Controller, GUI, ALV) gibt es eine extra Klasse mit dem Namen /CAMELOT/CL_OC_RRP, welche für die Business Add-In (BAI)s der RRP3 genutzt wird. Außerdem gibt es neben der Funktionsgruppe welche für das Hauptprogramm genutzt wird eine weiter Funktionsgruppe /CAMELOT/OC_COMMENT mit der Methode /CAMELOT/OC_COMMENT_SYNC welche Remote-Enabled ist. Das bedeutet sie kann aus einem anderen System heraus aufgerufen werden. Diese Methode hat als import paramter eine Tabelle mit dem Tabellentyp /CAMELOT/OC_ORD_COMMENT_RFC_T (siehe Anhang A.6: Dictionary-Objekte auf Seite vii). 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 Funktions Modules /SAPAPO/DM_ORDER_GET_ORDID die richtige orderid 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 Business Add-In (**BAdI**) verwendet. Der Unterschied zwischen einem Customer-Exits und einem Business Add-In (**BAdI**) ist, dass der Business Add-In (**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 Business Add-In (**BAdI**) /SAPAPO/RRP_IO_COL benötigt, um an den Feldkatalog der RRP3 das Feld Order_Comment anzufügen und diese dann auch mittels einer zweiten Methode zu befüllen. Außerdem musste eine Implicites Enhancement angelegt werden, da es mithilfe von Customer-Exits oder Business Add-In (**BAdI**)s nicht möglich war die t-style tabelle der einzelnen Elemente in der **ALV** Tabelle zu ändern, was allerdings absolut notwendig ist um z.B. den Kommentar editierbar zu machen. Die jeweiligen Methoden des Business Add-In (**BAdI**)s und des Impliciten Enhancements rufen eine statische Methode in meiner RRP3 Klasse auf, welche genau den selben Namen hat wie die eigentliche Methode, um sie später gut bei Kunden einzubinden, welche dann wiederum dann wiederum die Instance des Controllers bekommt und eine Methode aufruft.

5.11 Zwischenstand

Tabelle 7 zeigt den Zwischenstand nach der Implementierungsphase.

Vorgang	Geplant	Tatsächlich	Differenz
1. Anlegen der Datenbank	1 h	1 h	
2. Umsetzung der HTML-Oberflächen und Stylesheets	4 h	3 h	-1 h
3. Programmierung der PHP-Module für die Funktionen	23 h	23 h	
4. Nächtlichen Batchjob einrichten	1 h	1 h	

Tabelle 7: Zwischenstand nach der Implementierungsphase

5.12 Code Reviews während des Projekts

5.13 Manuelle Tests

5.14 Bug Fixing

6 Fazit

6.1 Soll-/Ist-Vergleich

6.2 Ausblick

- Wie wird sich das Projekt in Zukunft weiterentwickeln (z. B. geplante Erweiterungen)?

Literaturverzeichnis

Eidesstattliche Erklärung

Ich, Thomas Pöhlmann, versichere hiermit, dass ich meine **Dokumentation zur betrieblichen Projektarbeit** mit dem Thema

Order Comment Tool – Tool zu Unterstützung der Planer

selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, wobei ich alle wörtlichen und sinngemäßen Zitate als solche gekennzeichnet habe. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Mannheim, den 15.12.2018

THOMAS PÖHLMANN

A Anhang

A.1 Detaillierte Zeitplanung

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
1.2.1. Stati erstellen	0.25 h
1.2.2. Titel erstellen	0.25 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	0.5h
2.2.2. Stati erstellen	0.25h
2.2.3. Titel erstellen	0.25h
2.3 Implementierung der Klassen	6
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 Lastenheft (Auszug)

Es folgt ein Auszug aus dem Lastenheft mit Fokus auf die Anforderungen:

Die Anwendungen müssen folgende Anforderungen erfüllen:

1. Darstellung der Daten

- 1.1. Die Anwendung muss eine die Funktion bieten die Aufträge zu nach vorgegebenen Kriterien zu filter.
- 1.2. Die Aufträge müssen in einer, für den Planer guten Umgebung, dargestellt werden.
- 1.3. Die Aufträge sollen übersichtlich dargestellt werden.
- 1.4. Der Planer soll die möglichst einfach und schnell Kommentare schreiben und anzeigen können.
- 1.5. Der Planer soll zudem auch in den gewohnten standard Anwendungen Kommentare verfas-
sen und Lesen können

2. Weitere Anforderungen

- 2.1. Die Anwendungen soll sauber und übersichtlich Programmiert werden, sodass spätere Er-
weiterungen leicht durchzuführen sind.
- 2.2. Möglichst wenig Code soll in Userexits oder sonsitgen Erweiterungspunkten geschrieben
werden, da dieser immer per Hand beim Kunden eingefügt werden muss.

A.3 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 vorhandene 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.4 Use Case-Diagramm

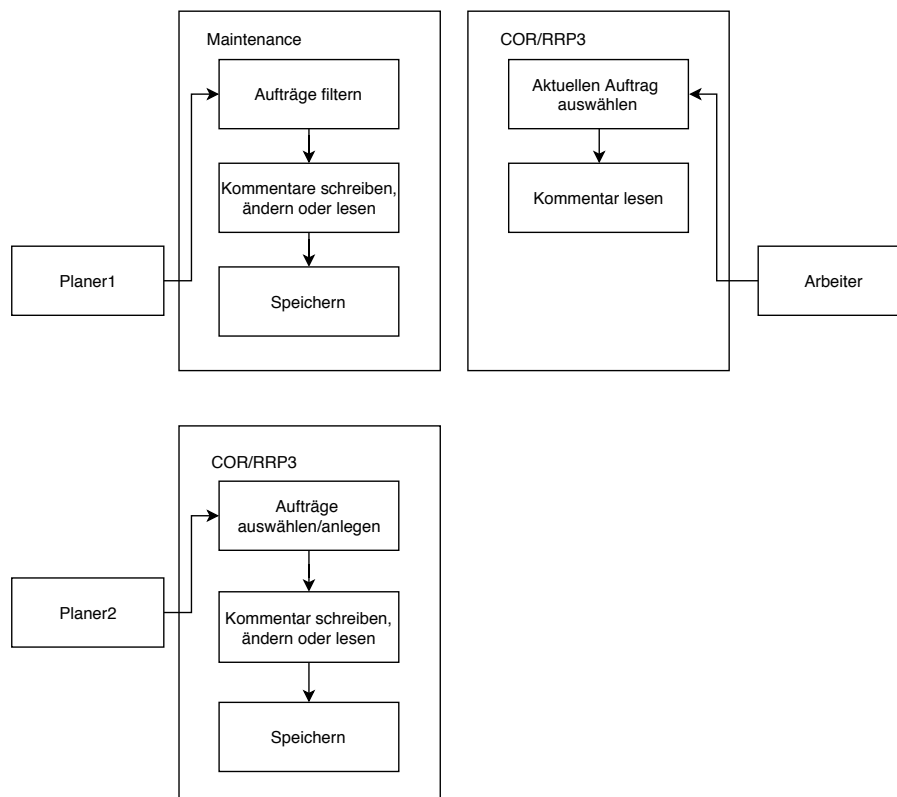


Abbildung 4: Use Case-Diagramm

A.5 Pflichtenheft (Auszug)

Zielbestimmung

1. Musskriterien

1.1. Modul-Liste: Zeigt eine filterbare Liste der Module mit den dazugehörigen Kerninformationen sowie Symbolen zur Einhaltung des Entwicklungsprozesses an

- In der Liste wird der Name, die Bibliothek und Daten zum Source und Kompilat eines Moduls angezeigt.
- Ebenfalls wird der Status des Moduls hinsichtlich Source und Kompilat angezeigt. Dazu gibt es unterschiedliche Status-Zeichen, welche symbolisieren in wie weit der Entwicklungsprozess eingehalten wurde bzw. welche Schritte als nächstes getan werden müssen. So gibt es z. B. Zeichen für das Einhalten oder Verletzen des Prozesses oder den Hinweis auf den nächsten zu tätigenden Schritt.
- Weiterhin werden die Benutzer und Zeitpunkte der aktuellen Version der Sourcen und Kompilate angezeigt. Dazu kann vorher ausgewählt werden, von welcher Umgebung diese Daten gelesen werden sollen.

A Anhang

- Es kann eine Filterung nach allen angezeigten Daten vorgenommen werden. Die Daten zu den Sourcen sind historisiert. Durch die Filterung ist es möglich, auch Module zu finden, die in der Zwischenzeit schon von einem anderen Benutzer editiert wurden.
- 1.2. Tag-Liste: Bietet die Möglichkeit die Module anhand von Tags zu filtern.
- Es sollen die Tags angezeigt werden, nach denen bereits gefiltert wird und die, die noch der Filterung hinzugefügt werden könnten, ohne dass die Ergebnisliste leer wird.
 - Zusätzlich sollen die Module angezeigt werden, die den Filterkriterien entsprechen. Sollten die Filterkriterien leer sein, werden nur die Module angezeigt, welche mit einem Tag versehen sind.
- 1.3. Import der Moduldaten aus einer bereitgestellten **CSV!**-Datei
- Es wird täglich eine Datei mit den Daten der aktuellen Module erstellt. Diese Datei wird (durch einen Cronjob) automatisch nachts importiert.
 - Dabei wird für jedes importierte Modul ein Zeitstempel aktualisiert, damit festgestellt werden kann, wenn ein Modul gelöscht wurde.
 - Die Datei enthält die Namen der Umgebung, der Bibliothek und des Moduls, den Programmtyp, den Benutzer und Zeitpunkt des Sourcecodes sowie des Kompilats und den Hash des Sourcecodes.
 - Sollte sich ein Modul verändert haben, werden die entsprechenden Daten in der Datenbank aktualisiert. Die Veränderungen am Source werden dabei aber nicht ersetzt, sondern historisiert.
- 1.4. Import der Informationen aus **SVN!** (**SVN!**). Durch einen „post-commit-hook“ wird nach jedem Einchecken eines Moduls ein **PHP!**-Script auf der Konsole aufgerufen, welches die Informationen, die vom **SVN!**-Kommandozeilentool geliefert werden, an **NatInfo!** übergibt.
- 1.5. Parsen der Sourcen
- Die Sourcen der Entwicklungsumgebung werden nach Tags, Links zu Artikeln im Wiki und Programmbeschreibungen durchsucht.
 - Diese Daten werden dann entsprechend angelegt, aktualisiert oder nicht mehr gesetzte Tags/Wikiartikel entfernt.
- 1.6. Sonstiges
- Das Programm läuft als Webanwendung im Intranet.
 - Die Anwendung soll möglichst leicht erweiterbar sein und auch von anderen Entwicklungsprozessen ausgehen können.
 - Eine Konfiguration soll möglichst in zentralen Konfigurationsdateien erfolgen.

Produkteinsatz

1. Anwendungsbereiche

Die Webanwendung dient als Anlaufstelle für die Entwicklung. Dort sind alle Informationen

A Anhang

für die Module an einer Stelle gesammelt. Vorher getrennte Anwendungen werden ersetzt bzw. verlinkt.

2. Zielgruppen

NatInfo wird lediglich von den **Natural!** (**Natural!**)-Entwicklern in der EDV-Abteilung genutzt.

3. Betriebsbedingungen

Die nötigen Betriebsbedingungen, also der Webserver, die Datenbank, die Versionsverwaltung, das Wiki und der nächtliche Export sind bereits vorhanden und konfiguriert. Durch einen täglichen Cronjob werden entsprechende Daten aktualisiert, die Webanwendung ist jederzeit aus dem Intranet heraus erreichbar.

A.6 Dictionary-Objekte

A.6.1 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.6.2 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

A Anhang

	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.6.3 Datenelemente

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

A.6.4 Domänen

Name	Date Type
/camelot/oc_ord_comment	char60

A.7 Screenshots der Anwendung

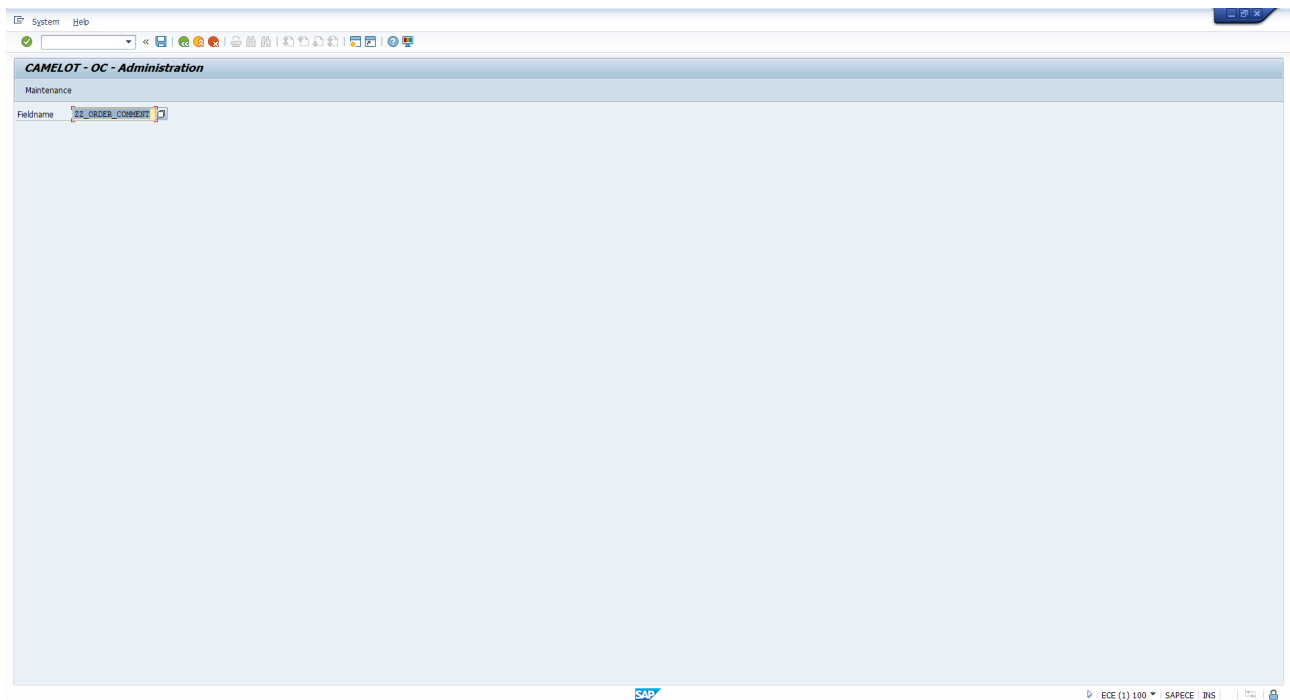


Abbildung 5: Ändern des Feldnamens für den Kommentar in der AUFK Datenbanktabelle

A Anhang

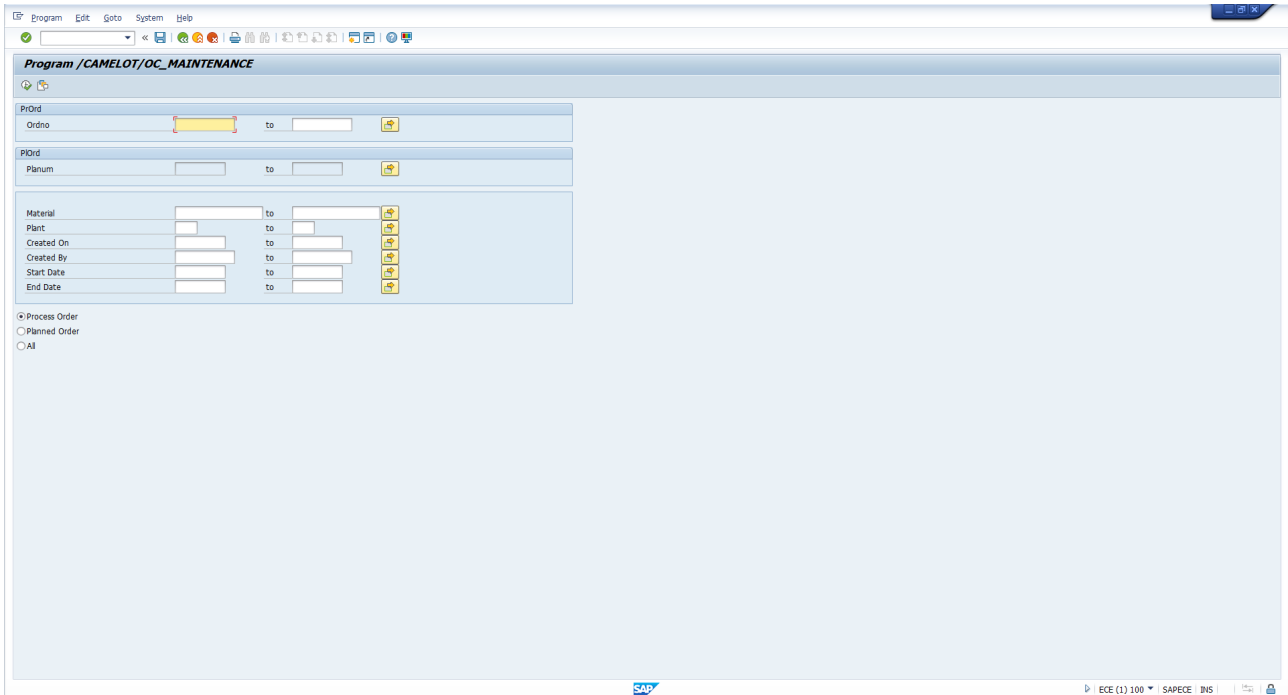
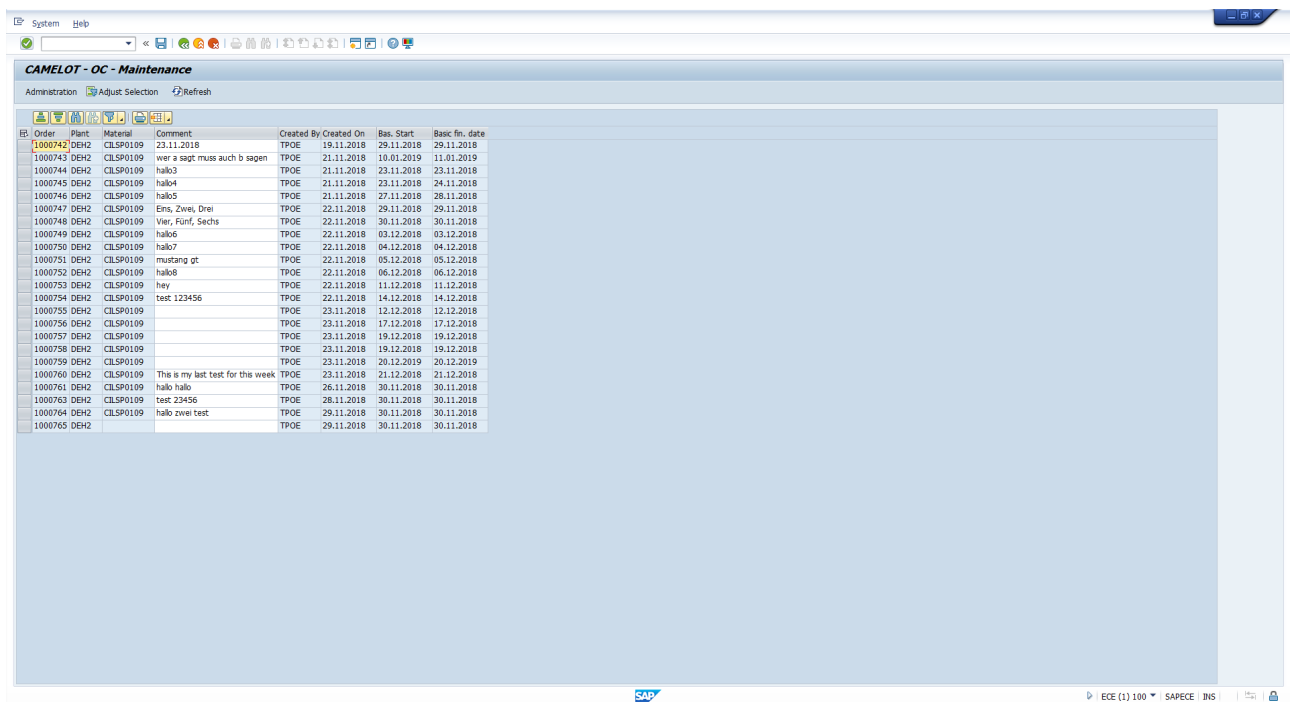


Abbildung 6: Selektionsbildschirm für den Maintenance Screen

A Anhang

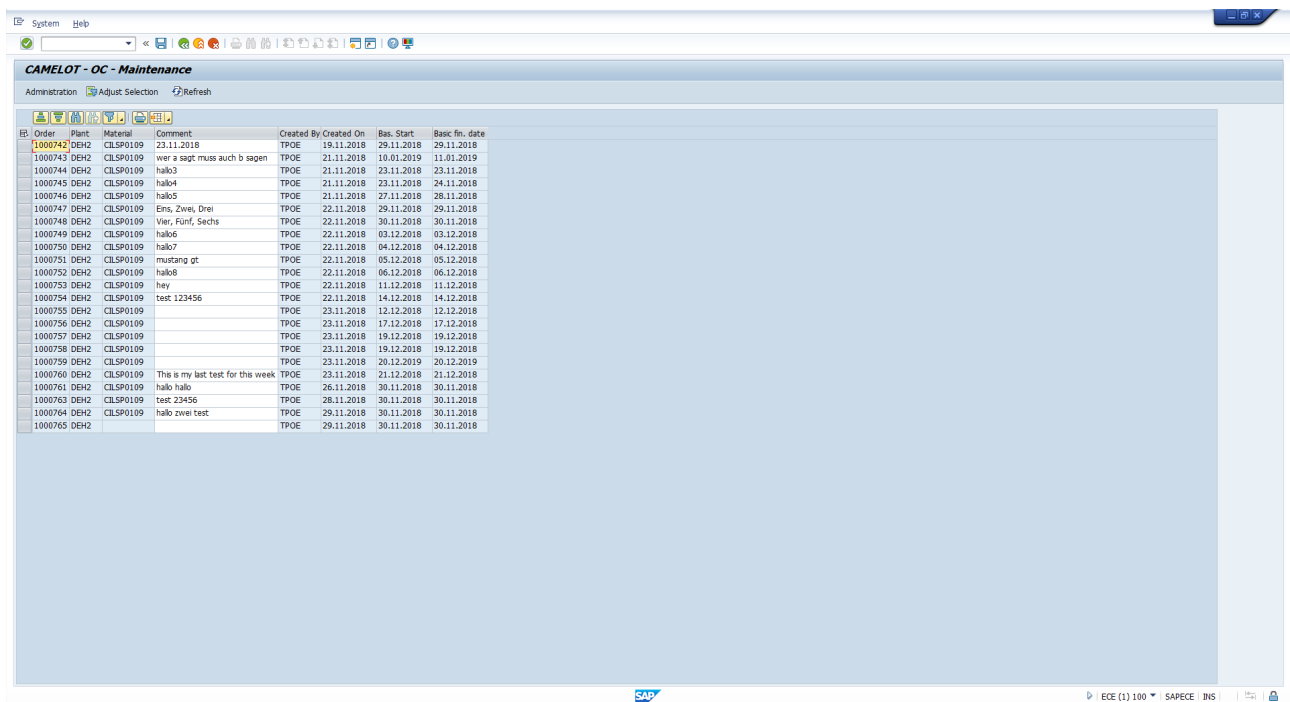


The screenshot displays the 'CAMELOT - OC - Maintenance' application window. It features a menu bar with 'System' and 'Help', a toolbar with various icons, and a main table of maintenance orders. The table has columns for Order, Plant, Material, Comment, Created By, Created On, Bas. Start, and Basic fn. date. The data is sorted by Order number, ranging from 1000742 to 1000765. The interface also includes a status bar at the bottom with 'SAP' and 'ECE (1) 100'.

Order	Plant	Material	Comment	Created By	Created On	Bas. Start	Basic fn. date
1000742	DEH2	CLSP0109	23.11.2018	TPOE	19.11.2018	29.11.2018	29.11.2018
1000743	DEH2	CLSP0109	wer a sagt muss auch b sagen	TPOE	21.11.2018	10.01.2019	11.01.2019
1000744	DEH2	CLSP0109	hallo3	TPOE	21.11.2018	23.11.2018	23.11.2018
1000745	DEH2	CLSP0109	hallo4	TPOE	21.11.2018	23.11.2018	24.11.2018
1000746	DEH2	CLSP0109	hallo5	TPOE	21.11.2018	27.11.2018	28.11.2018
1000747	DEH2	CLSP0109	Eins, Zwei, Drei	TPOE	22.11.2018	29.11.2018	29.11.2018
1000748	DEH2	CLSP0109	Vier, Fünf, Sechs	TPOE	22.11.2018	30.11.2018	30.11.2018
1000749	DEH2	CLSP0109	hallo6	TPOE	22.11.2018	03.12.2018	03.12.2018
1000750	DEH2	CLSP0109	hallo7	TPOE	22.11.2018	04.12.2018	04.12.2018
1000751	DEH2	CLSP0109	mustang gt	TPOE	22.11.2018	05.12.2018	05.12.2018
1000752	DEH2	CLSP0109	hallo8	TPOE	22.11.2018	06.12.2018	06.12.2018
1000753	DEH2	CLSP0109	hey	TPOE	22.11.2018	11.12.2018	11.12.2018
1000754	DEH2	CLSP0109	test 123456	TPOE	22.11.2018	14.12.2018	14.12.2018
1000755	DEH2	CLSP0109		TPOE	23.11.2018	12.12.2018	12.12.2018
1000756	DEH2	CLSP0109		TPOE	23.11.2018	17.12.2018	17.12.2018
1000757	DEH2	CLSP0109		TPOE	23.11.2018	19.12.2018	19.12.2018
1000758	DEH2	CLSP0109		TPOE	23.11.2018	19.12.2018	19.12.2018
1000759	DEH2	CLSP0109		TPOE	23.11.2018	20.12.2019	20.12.2019
1000760	DEH2	CLSP0109	This is my last test for this week	TPOE	23.11.2018	21.12.2018	21.12.2018
1000761	DEH2	CLSP0109	hallo hallo	TPOE	26.11.2018	30.11.2018	30.11.2018
1000763	DEH2	CLSP0109	test 23456	TPOE	28.11.2018	30.11.2018	30.11.2018
1000764	DEH2	CLSP0109	hallo zwei test	TPOE	29.11.2018	30.11.2018	30.11.2018
1000765	DEH2	CLSP0109		TPOE	29.11.2018	30.11.2018	30.11.2018

Abbildung 7:

A Anhang



The screenshot displays the SAP 'CAMELOT - OC - Maintenance' interface. It features a standard SAP menu bar at the top with 'System' and 'Help' options. Below the menu is a toolbar with various icons for navigation and actions. The main area contains a table with the following columns: Order, Plant, Material, Comment, Created By, Created On, Bas. Start, and Basic fn. date. The table lists 20 maintenance orders, each with a unique order number, plant code (DEH2), material code (CLSP0109), a descriptive comment, and associated dates. The interface also includes a sidebar on the left with icons for different views and a status bar at the bottom showing 'ECE (1) 100' and 'SAPECE INS'.

Order	Plant	Material	Comment	Created By	Created On	Bas. Start	Basic fn. date
1000742	DEH2	CLSP0109	23.11.2018	TPOE	19.11.2018	29.11.2018	29.11.2018
1000743	DEH2	CLSP0109	wer a sagt muss auch b sagen	TPOE	21.11.2018	10.01.2019	11.01.2019
1000744	DEH2	CLSP0109	hallo3	TPOE	21.11.2018	23.11.2018	23.11.2018
1000745	DEH2	CLSP0109	hallo4	TPOE	21.11.2018	23.11.2018	24.11.2018
1000746	DEH2	CLSP0109	hallo5	TPOE	21.11.2018	27.11.2018	28.11.2018
1000747	DEH2	CLSP0109	Eins, Zwei, Drei	TPOE	22.11.2018	29.11.2018	29.11.2018
1000748	DEH2	CLSP0109	Vier, Fünf, Sechs	TPOE	22.11.2018	30.11.2018	30.11.2018
1000749	DEH2	CLSP0109	hallo6	TPOE	22.11.2018	03.12.2018	03.12.2018
1000750	DEH2	CLSP0109	hallo7	TPOE	22.11.2018	04.12.2018	04.12.2018
1000751	DEH2	CLSP0109	mustang gt	TPOE	22.11.2018	05.12.2018	05.12.2018
1000752	DEH2	CLSP0109	hallo8	TPOE	22.11.2018	06.12.2018	06.12.2018
1000753	DEH2	CLSP0109	hey	TPOE	22.11.2018	11.12.2018	11.12.2018
1000754	DEH2	CLSP0109	test 123456	TPOE	22.11.2018	14.12.2018	14.12.2018
1000755	DEH2	CLSP0109		TPOE	23.11.2018	12.12.2018	12.12.2018
1000756	DEH2	CLSP0109		TPOE	23.11.2018	17.12.2018	17.12.2018
1000757	DEH2	CLSP0109		TPOE	23.11.2018	19.12.2018	19.12.2018
1000758	DEH2	CLSP0109		TPOE	23.11.2018	19.12.2018	19.12.2018
1000759	DEH2	CLSP0109		TPOE	23.11.2018	20.12.2019	20.12.2019
1000760	DEH2	CLSP0109	This is my last test for this week	TPOE	23.11.2018	21.12.2018	21.12.2018
1000761	DEH2	CLSP0109	hallo hallo	TPOE	26.11.2018	30.11.2018	30.11.2018
1000763	DEH2	CLSP0109	test 23456	TPOE	28.11.2018	30.11.2018	30.11.2018
1000764	DEH2	CLSP0109	hallo zwei test	TPOE	29.11.2018	30.11.2018	30.11.2018
1000765	DEH2	CLSP0109		TPOE	29.11.2018	30.11.2018	30.11.2018

Abbildung 8:

A Anhang

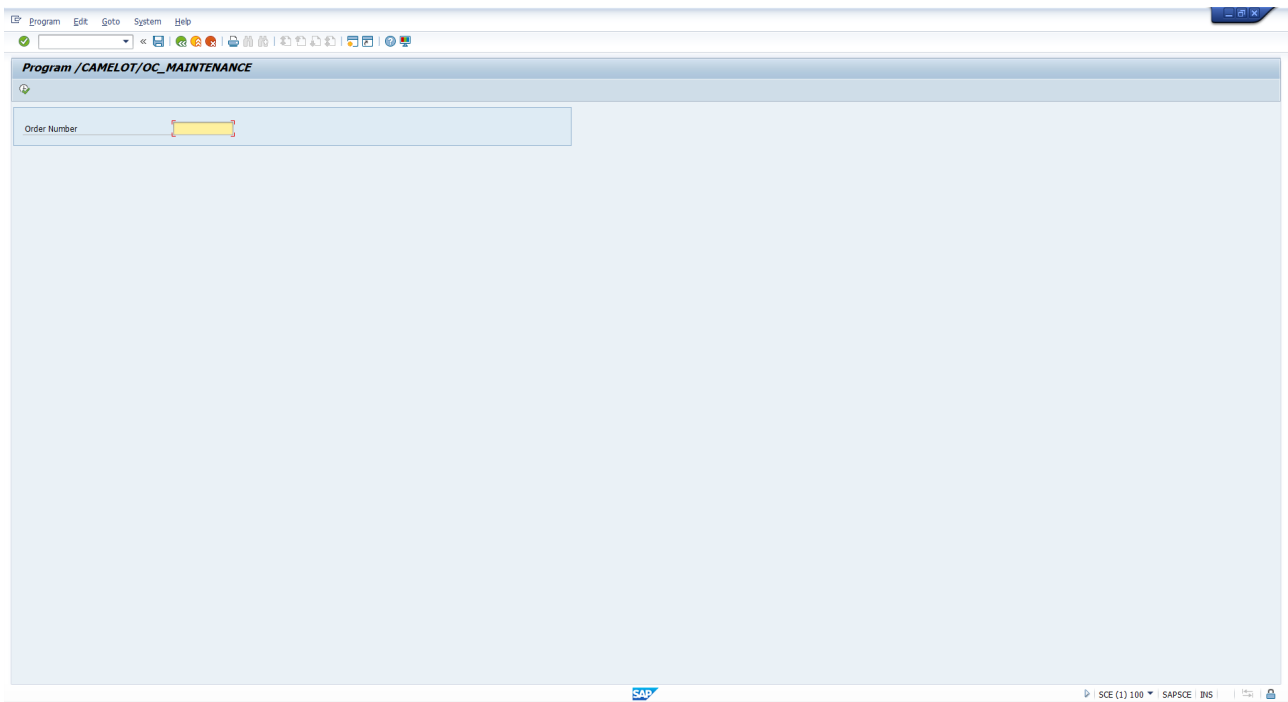


Abbildung 9:

A Anhang

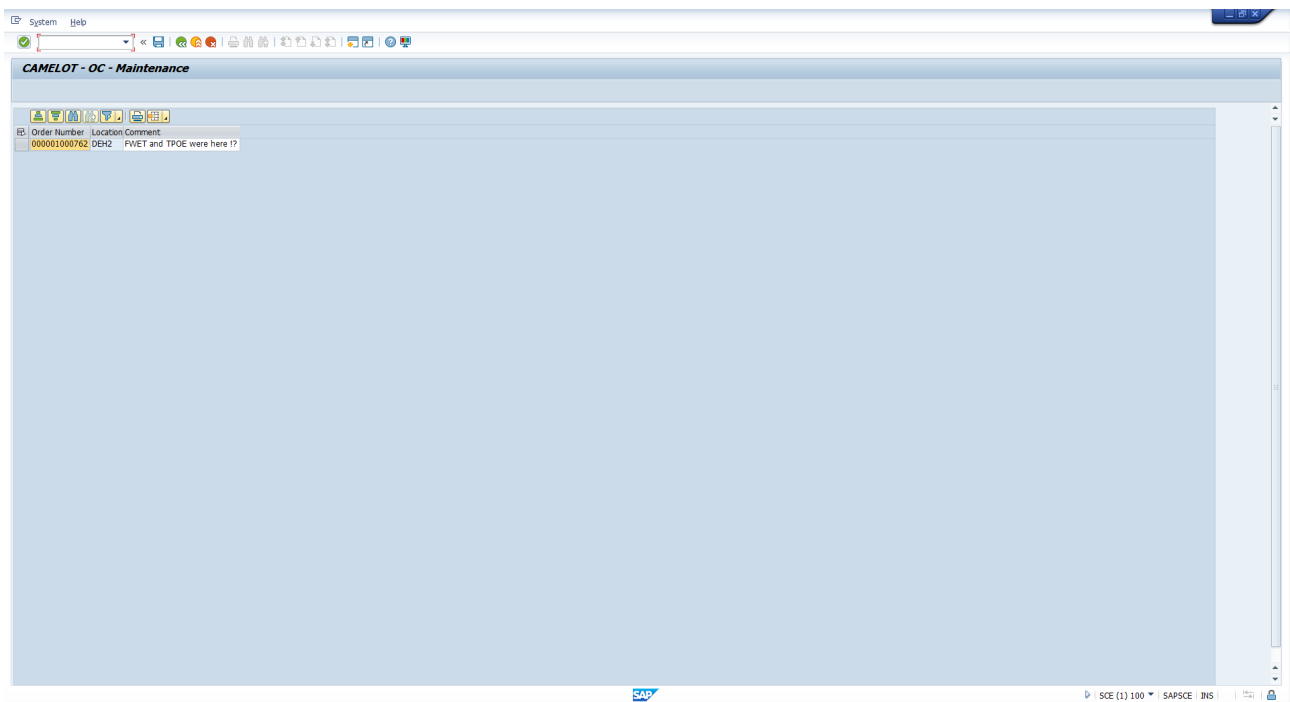


Abbildung 10:

ORDER COMMENT TOOL

Tool zu Unterstützung der Planer

A Anhang

Interactive Planning

Edit

Goto

Settings

DS Planning Board

System

Help

Abbildung 11:

A Anhang

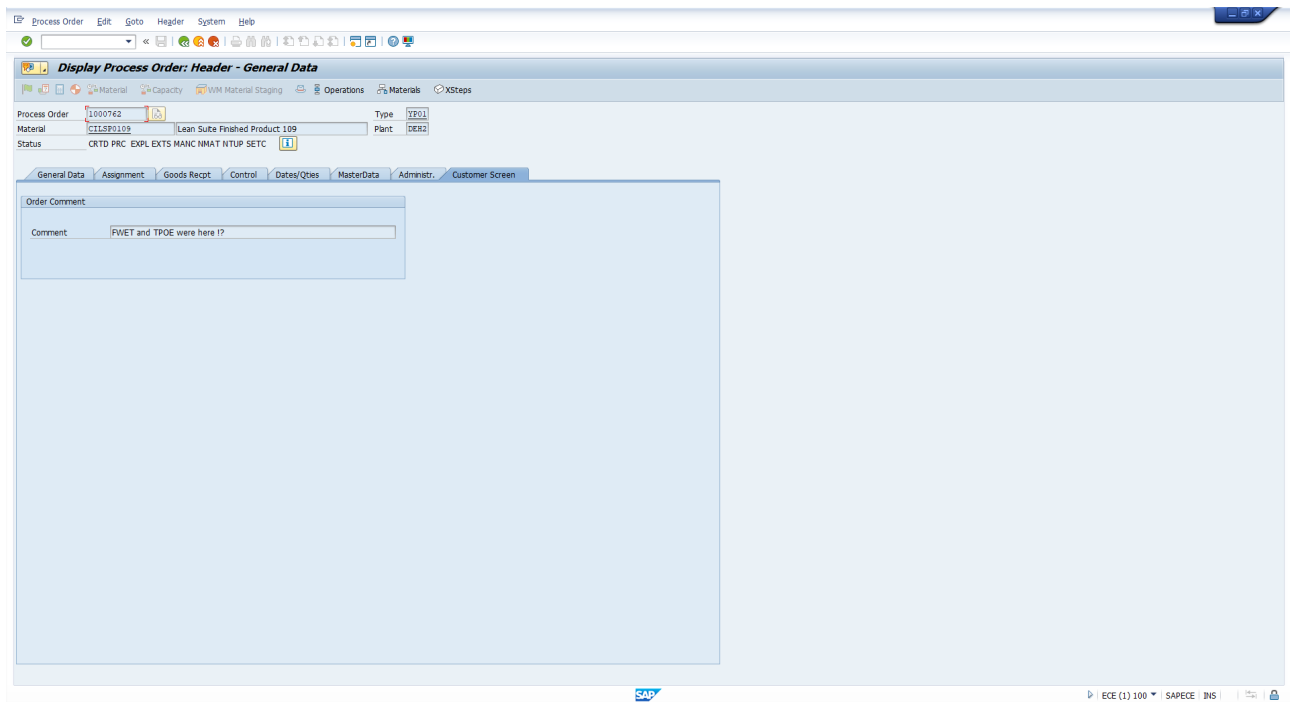


Abbildung 13:

A Anhang

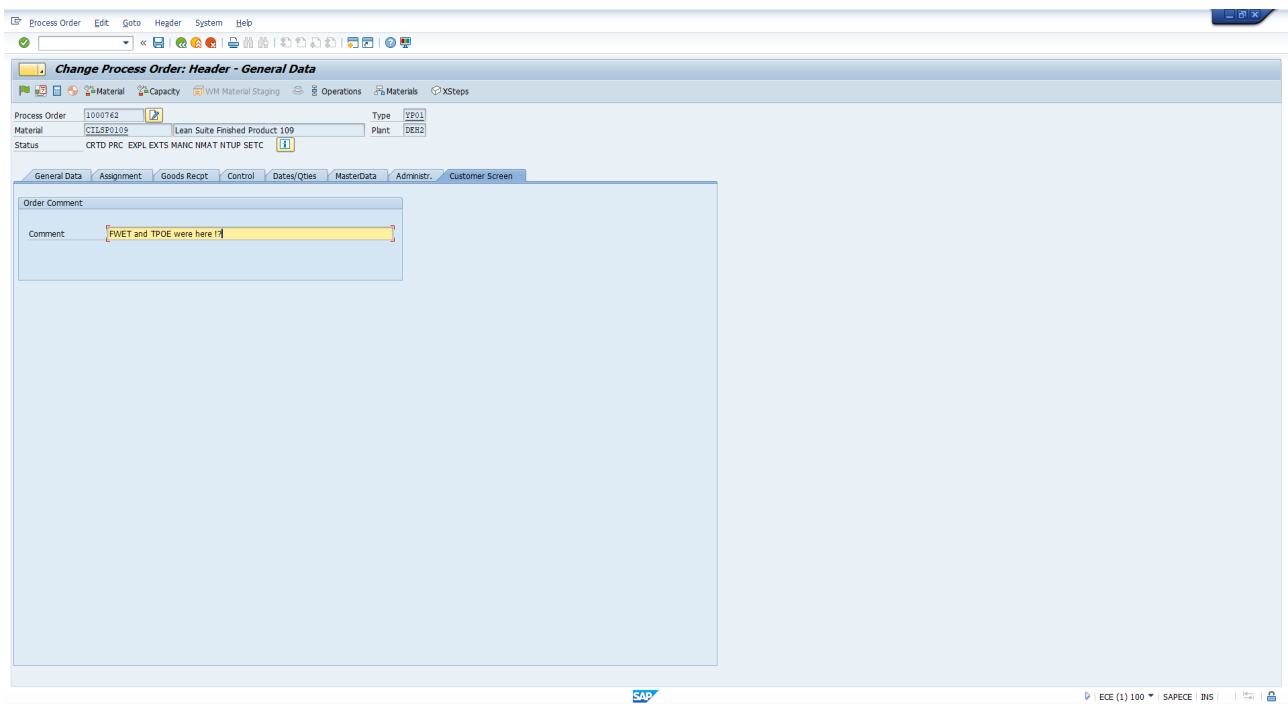


Abbildung 14:

A.8 Programming Guidlines

A.9 Klassendiagramm

Klassendiagramme und weitere UML-Diagramme kann man auch direkt mit \LaTeX zeichnen, siehe z. B. <http://metauml.sourceforge.net/old/class-diagram.html>.

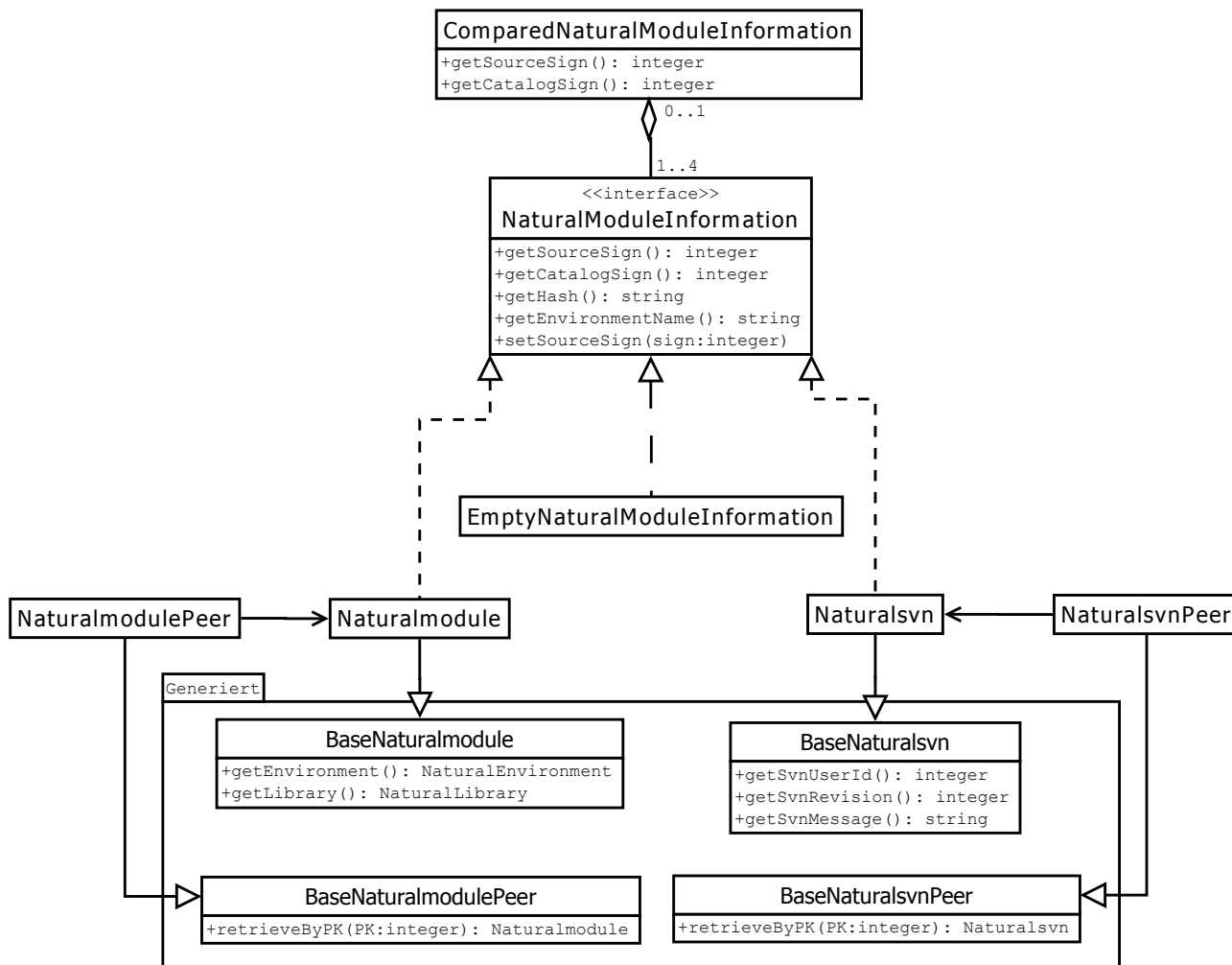


Abbildung 15: Klassendiagramm