

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	IV
Tabellenverzeichnis	V
Listings	VI
Abkürzungsverzeichnis	VII
Transaktionsverzeichnis	VIII
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 Geschäftslogik	9
4.6 Maßnahmen zur Qualitätssicherung	9

Inhaltsverzeichnis

4.7	Pflichtenheft/Datenverarbeitungskonzept	10
4.8	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	11
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	13
5.11	Zwischenstand	13
6	Abnahmephase	13
6.1	Zwischenstand	13
7	Einführungsphase	14
7.1	Zwischenstand	14
8	Dokumentation	14
8.1	Zwischenstand	14
9	Fazit	15
9.1	Soll-/Ist-Vergleich	15
9.2	Lessons Learned	15
9.3	Ausblick	16
	Literaturverzeichnis	17
	Eidesstattliche Erklärung	18
A	Anhang	i
A.1	Detaillierte Zeitplanung	i
A.2	Lastenheft (Auszug)	ii
A.3	Use Case-Diagramm	iii
A.4	Pflichtenheft (Auszug)	iii
A.5	Datenbankmodell	v
A.6	Oberflächenentwürfe	vi
A.7	Screenshots der Anwendung	viii
A.8	Entwicklerdokumentation	x
A.9	Testfall und sein Aufruf auf der Konsole	xii

Inhaltsverzeichnis

A.10	Klasse: ComparedNaturalModuleInformation	xiii
A.11	Klassendiagramm	xvi
A.12	Benutzerdokumentation	xvii

Abbildungsverzeichnis

1	Vereinfachtes ER-Modell	9
2	Prozess des Einlesens eines Moduls	10
3	Use Case-Diagramm	iii
4	Datenbankmodell	v
5	Liste der Module mit Filtermöglichkeiten	vi
6	Anzeige der Übersichtsseite einzelner Module	vii
7	Anzeige und Filterung der Module nach Tags	vii
8	Anzeige und Filterung der Module nach Tags	viii
9	Liste der Module mit Filtermöglichkeiten	ix
10	Aufruf des Testfalls auf der Konsole	xiii
11	Klassendiagramm	xvi

Tabellenverzeichnis

1	BeteiligtePersonen	2
2	Zeitplanung	3
3	HardwareKosten	5
4	ProjektKosten	5
5	Zwischenstand nach der Analysephase	7
6	Zwischenstand nach der Entwurfsphase	10
7	Zwischenstand nach der Implementierungsphase	13
8	Zwischenstand nach der Abnahmephase	13
9	Zwischenstand nach der Einführungsphase	14
10	Zwischenstand nach der Dokumentation	15
11	Soll-/Ist-Vergleich	15

Listings

1	Testfall in PHP	xii
2	Klasse: ComparedNaturalModuleInformation	xiii

Abkürzungsverzeichnis

ECC	ERP Central Component
PBO	Process Before Output
PAI	Process After Input
ALV	ABAP List View
ERM	Entity-Relationship-Modell
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 APO ("Advanced Planning & Optimization") 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

Da wir bereits ein großes Portfolio an verschiedenen Tools im SAP Umfeld haben, welche den Planer unterstützen

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€.

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

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

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

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

3 Analysephase

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

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

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

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

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

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 Amortisations-

3 Analysephase

dauer wird berechnet indem man die Produktionskosten bzw. Anschaffungskosten durch die Gewinne dividiert, welche durch das neue Produkt erzielt wurden.

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

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

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 $\frac{500 \text{ €}}{50 \cdot 2300 \text{ €/Jahr}} \approx 0,004 \text{ Jahre} \approx 1,5 \text{ Tage}$.

3 Analysephase

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

z.b Administration nicht in key felder der tabelle schreiben input checks.

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	3 h	4 h	+1 h
2. „Make or buy“-Entscheidung und Wirtschaftlichkeitsanalyse	1 h	1 h	
3. Erstellen eines „Use-Case“-Diagramms	2 h	2 h	
4. 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 ECC werden nach dem MVC (Model View Controller) Konzept programmiert. Allerdings habe ich in diesem Fall kein Model benötigt. Es gibt in jedem System jeweils eine 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 PBO (Process Before Output) und PAI (Process After Input) Methode. Die jeweiligen Screens werden in einer Funktionsgruppe definiert und mithilfe des SAP Screen Painters 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 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 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/ORDFLDS gespeichert werden.

4.4 Datenmodell

- Entwurf/Beschreibung der Datenstrukturen (z. B. [ERM](#) und/oder Tabellenmodell, **XML!**-Schemas) mit kurzer Beschreibung der wichtigsten (!) verwendeten Entitäten.

Beispiel In [Abbildung 1](#) wird ein Entity-Relationship-Modell ([ERM](#)) dargestellt, welches lediglich Entitäten, Relationen und die dazugehörigen Kardinalitäten enthält.

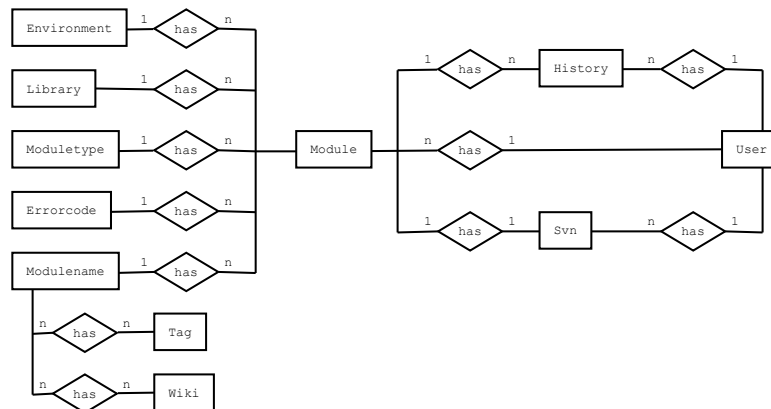


Abbildung 1: Vereinfachtes ER-Modell

4.5 Geschäftslogik

- Modellierung und Beschreibung der wichtigsten (!) Bereiche der Geschäftslogik (z. B. mit Komponenten-, Klassen-, Sequenz-, Datenflussdiagramm, Programmablaufplan, Struktogramm, **EPK!** (**EPK!**)).
- Wie wird die erstellte Anwendung in den Arbeitsfluss des Unternehmens integriert?

Beispiel Ein Klassendiagramm, welches die Klassen der Anwendung und deren Beziehungen untereinander darstellt kann im Anhang [A.11: Klassendiagramm](#) auf Seite [xvi](#) eingesehen werden.

[Abbildung 2](#) zeigt den grundsätzlichen Programmablauf beim Einlesen eines Moduls als **EPK!**.

4.6 Maßnahmen zur Qualitätssicherung

Um die hohen Qualitätsanforderungen der Camelot ITLab zu gewährleisten und die Qualitätsanforderungen des Projekts zu sicher, 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 Entwurfsphase

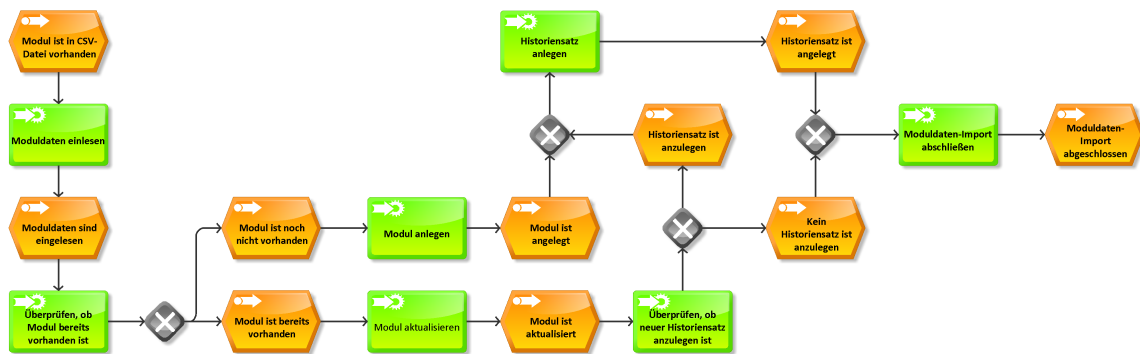


Abbildung 2: Prozess des Einlesens eines Moduls

4.7 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.4: [Pflichtenheft \(Auszug\)](#) auf Seite iii zu finden.

4.8 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.

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 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 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 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 im Anhang auf Seite XX.

Beispiel Screenshots der Anwendung in der Entwicklungsphase mit Dummy-Daten befinden sich im Anhang [A.7: Screenshots der Anwendung](#) auf Seite [viii](#).

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

5 Implementierungsphase

Struktur des jeweiligen Screens ist, Initial also "leer". 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 alle 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.

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. Dieses 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 Eingabefeld 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 einem späteren Zeitpunkt dann der RFC ausgeführt wird.

5.7 Implementierung der Datenstruktur APO

5.8 Implementierung der Benutzeroberfläche APO

5.9 Implementierung der Geschäftslogik APO

5.10 Implementierung der RRP3 Erweiterung

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

6 Abnahmephase

- Welche Tests (z. B. Unit-, Integrations-, Systemtests) wurden durchgeführt und welche Ergebnisse haben sie geliefert (z. B. Logs von Unit Tests, Testprotokolle der Anwender)?
- Wurde die Anwendung offiziell abgenommen?

Beispiel Ein Auszug eines Unit Tests befindet sich im Anhang [A.9: Testfall und sein Aufruf auf der Konsole](#) auf Seite [xii](#). Dort ist auch der Aufruf des Tests auf der Konsole des Webserverns zu sehen.

6.1 Zwischenstand

Tabelle 8 zeigt den Zwischenstand nach der Abnahmephase.

Vorgang	Geplant	Tatsächlich	Differenz
1. Abnahmetest der Fachabteilung	1 h	1 h	

Tabelle 8: Zwischenstand nach der Abnahmephase

7 Einführungsphase

- Welche Schritte waren zum Deployment der Anwendung nötig und wie wurden sie durchgeführt (automatisiert/manuell)?
- Wurden ggfs. Altdaten migriert und wenn ja, wie?
- Wurden Benutzerschulungen durchgeführt und wenn ja, Wie wurden sie vorbereitet?

7.1 Zwischenstand

Tabelle 9 zeigt den Zwischenstand nach der Einführungsphase.

Vorgang	Geplant	Tatsächlich	Differenz
1. Einführung/Benutzerschulung	1 h	1 h	

Tabelle 9: Zwischenstand nach der Einführungsphase

8 Dokumentation

- Wie wurde die Anwendung für die Benutzer/Administratoren/Entwickler dokumentiert (z. B. Benutzerhandbuch, **API!**-Dokumentation)?
- Hinweis: Je nach Zielgruppe gelten bestimmte Anforderungen für die Dokumentation (z. B. keine IT-Fachbegriffe in einer Anwenderdokumentation verwenden, aber auf jeden Fall in einer Dokumentation für den IT-Bereich).

Beispiel Ein Ausschnitt aus der erstellten Benutzerdokumentation befindet sich im Anhang [A.12: Benutzerdokumentation](#) auf Seite [xvii](#). Die Entwicklerdokumentation wurde mittels PHPDoc¹ automatisch generiert. Ein beispielhafter Auszug aus der Dokumentation einer Klasse findet sich im Anhang [A.8: Entwicklerdokumentation](#) auf Seite [x](#).

8.1 Zwischenstand

Tabelle 10 zeigt den Zwischenstand nach der Dokumentation.

¹Vgl. PHPDOC.ORG [2010]

9 Fazit

Vorgang	Geplant	Tatsächlich	Differenz
1. Erstellen der Benutzerdokumentation	2 h	2 h	
2. Erstellen der Projektdokumentation	6 h	8 h	+2 h
3. Programmdokumentation	1 h	1 h	

Tabelle 10: Zwischenstand nach der Dokumentation

9 Fazit

9.1 Soll-/Ist-Vergleich

- Wurde das Projektziel erreicht und wenn nein, warum nicht?
- Ist der Auftraggeber mit dem Projektergebnis zufrieden und wenn nein, warum nicht?
- Wurde die Projektplanung (Zeit, Kosten, Personal, Sachmittel) eingehalten oder haben sich Abweichungen ergeben und wenn ja, warum?
- Hinweis: Die Projektplanung muss nicht strikt eingehalten werden. Vielmehr sind Abweichungen sogar als normal anzusehen. Sie müssen nur vernünftig begründet werden (z. B. durch Änderungen an den Anforderungen, unter-/überschätzter Aufwand).

Beispiel (verkürzt) Wie in Tabelle 11 zu erkennen ist, konnte die Zeitplanung bis auf wenige Ausnahmen eingehalten werden.

Phase	Geplant	Tatsächlich	Differenz
Entwurfsphase	19 h	19 h	
Analysephase	9 h	10 h	+1 h
Implementierungsphase	29 h	28 h	-1 h
Abnahmetest der Fachabteilung	1 h	1 h	
Einführungsphase	1 h	1 h	
Erstellen der Dokumentation	9 h	11 h	+2 h
Pufferzeit	2 h	0 h	-2 h
Gesamt	70 h	70 h	

Tabelle 11: Soll-/Ist-Vergleich

9.2 Lessons Learned

- Was hat der Prüfling bei der Durchführung des Projekts gelernt (z. B. Zeitplanung, Vorteile der eingesetzten Frameworks, Änderungen der Anforderungen)?

9.3 Ausblick

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

Literaturverzeichnis

phpdoc.org 2010

PHPDOC.ORG: *phpDocumentor-Website*. Version: 2010. <http://www.phpdoc.org/>, Abruf: 20.04.2010

Sensio Labs 2010

SENSIO LABS: *Symfony - Open-Source PHP Web Framework*. Version: 2010. <http://www.symfony-project.org/>, Abruf: 20.04.2010

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	9 h
1. Analyse des Ist-Zustands	3 h
1.1. Fachgespräch mit der EDV-Abteilung	1 h
1.2. Prozessanalyse	2 h
2. „Make or buy“-Entscheidung und Wirtschaftlichkeitsanalyse	1 h
3. Erstellen eines „Use-Case“-Diagramms	2 h
4. Erstellen des Lastenhefts mit der EDV-Abteilung	3 h
Entwurfsphase	19 h
1. Prozessentwurf	2 h
2. Datenbankentwurf	3 h
2.1. ER-Modell erstellen	2 h
2.2. Konkretes Tabellenmodell erstellen	1 h
3. Erstellen von Datenverarbeitungskonzepten	4 h
3.1. Verarbeitung der CSV-Daten	1 h
3.2. Verarbeitung der SVN-Daten	1 h
3.3. Verarbeitung der Sourcen der Programme	2 h
4. Benutzeroberflächen entwerfen und abstimmen	2 h
5. Erstellen eines UML-Komponentendiagramms der Anwendung	4 h
6. Erstellen des Pflichtenhefts	4 h
Implementierungsphase	29 h
1. Anlegen der Datenbank	1 h
2. Umsetzung der HTML-Oberflächen und Stylesheets	4 h
3. Programmierung der PHP-Module für die Funktionen	23 h
3.1. Import der Modulinformationen aus CSV-Dateien	2 h
3.2. Parsen der Modulquelltexte	3 h
3.3. Import der SVN-Daten	2 h
3.4. Vergleichen zweier Umgebungen	4 h
3.5. Abrufen der von einem zu wählenden Benutzer geänderten Module	3 h
3.6. Erstellen einer Liste der Module unter unterschiedlichen Aspekten	5 h
3.7. Anzeigen einer Liste mit den Modulen und geparsen Metadaten	3 h
3.8. Erstellen einer Übersichtsseite für ein einzelnes Modul	1 h
4. Nächtlichen Batchjob einrichten	1 h
Abnahmetest der Fachabteilung	1 h
1. Abnahmetest der Fachabteilung	1 h
Einführungsphase	1 h
1. Einführung/Benutzerschulung	1 h
Erstellen der Dokumentation	9 h
1. Erstellen der Benutzerdokumentation	2 h
2. Erstellen der Projektdokumentation	6 h
3. Programmdokumentation	1 h
3.1. Generierung durch PHPdoc	1 h
Pufferzeit	2 h
1. Puffer	2 h
Gesamt	70 h

A.2 Lastenheft (Auszug)

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

Die Anwendung muss folgende Anforderungen erfüllen:

1. Verarbeitung der Moduldaten
 - 1.1. Die Anwendung muss die von Subversion und einem externen Programm bereitgestellten Informationen (z.B. Source-Benutzer, -Datum, Hash) verarbeiten.
 - 1.2. Auslesen der Beschreibung und der Stichwörter aus dem Sourcecode.
2. Darstellung der Daten
 - 2.1. Die Anwendung muss eine Liste aller Module erzeugen inkl. Source-Benutzer und -Datum, letztem Commit-Benutzer und -Datum für alle drei Umgebungen.
 - 2.2. Verknüpfen der Module mit externen Tools wie z.B. Wiki-Einträgen zu den Modulen oder dem Sourcecode in Subversion.
 - 2.3. Die Sourcen der Umgebungen müssen verglichen und eine schnelle Übersicht zur Einhaltung des allgemeinen Entwicklungsprozesses gegeben werden.
 - 2.4. Dieser Vergleich muss auf die von einem bestimmten Benutzer bearbeiteten Module eingeschränkt werden können.
 - 2.5. Die Anwendung muss in dieser Liste auch Module anzeigen, die nach einer Bearbeitung durch den gesuchten Benutzer durch jemand anderen bearbeitet wurden.
 - 2.6. Abweichungen sollen kenntlich gemacht werden.
 - 2.7. Anzeigen einer Übersichtsseite für ein Modul mit allen relevanten Informationen zu diesem.
3. Sonstige Anforderungen
 - 3.1. Die Anwendung muss ohne das Installieren einer zusätzlichen Software über einen Webbrowser im Intranet erreichbar sein.
 - 3.2. Die Daten der Anwendung müssen jede Nacht bzw. nach jedem **SVN!**-Commit automatisch aktualisiert werden.
 - 3.3. Es muss ermittelt werden, ob Änderungen auf der Produktionsumgebung vorgenommen wurden, die nicht von einer anderen Umgebung kopiert wurden. Diese Modulliste soll als Mahnung per E-Mail an alle Entwickler geschickt werden (Peer Pressure).
 - 3.4. Die Anwendung soll jederzeit erreichbar sein.
 - 3.5. Da sich die Entwickler auf die Anwendung verlassen, muss diese korrekte Daten liefern und darf keinen Interpretationsspielraum lassen.
 - 3.6. Die Anwendung muss so flexibel sein, dass sie bei Änderungen im Entwicklungsprozess einfach angepasst werden kann.

A.3 Use Case-Diagramm

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

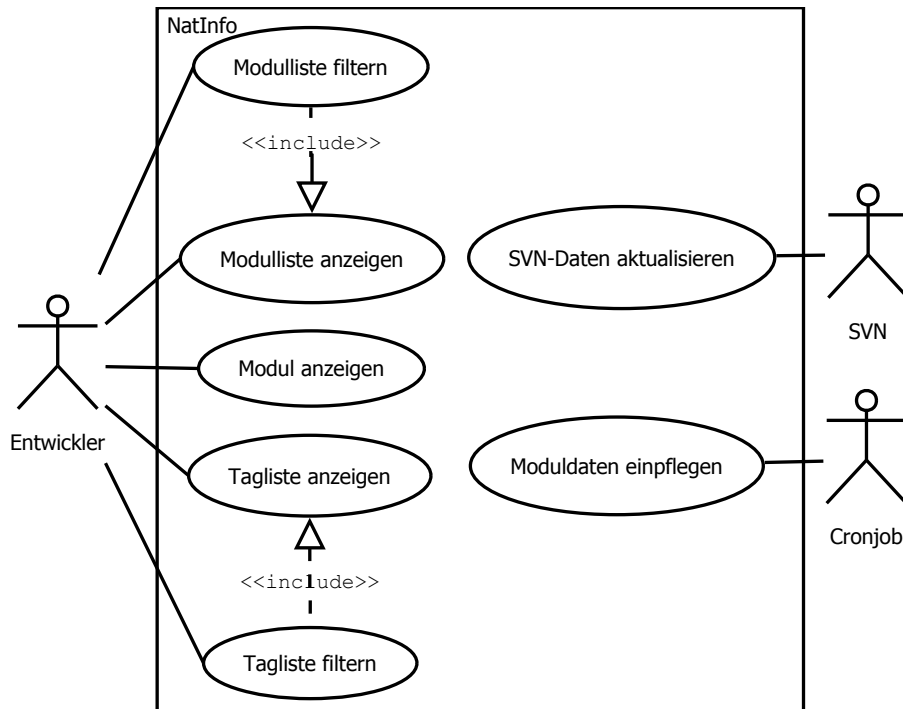


Abbildung 3: Use Case-Diagramm

A.4 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.

F

A.5 Datenbankmodell

ER-Modelle kann man auch direkt mit \LaTeX zeichnen, siehe z. B. <http://www.texample.net/tikz/examples/entity-relationship-diagram/>.

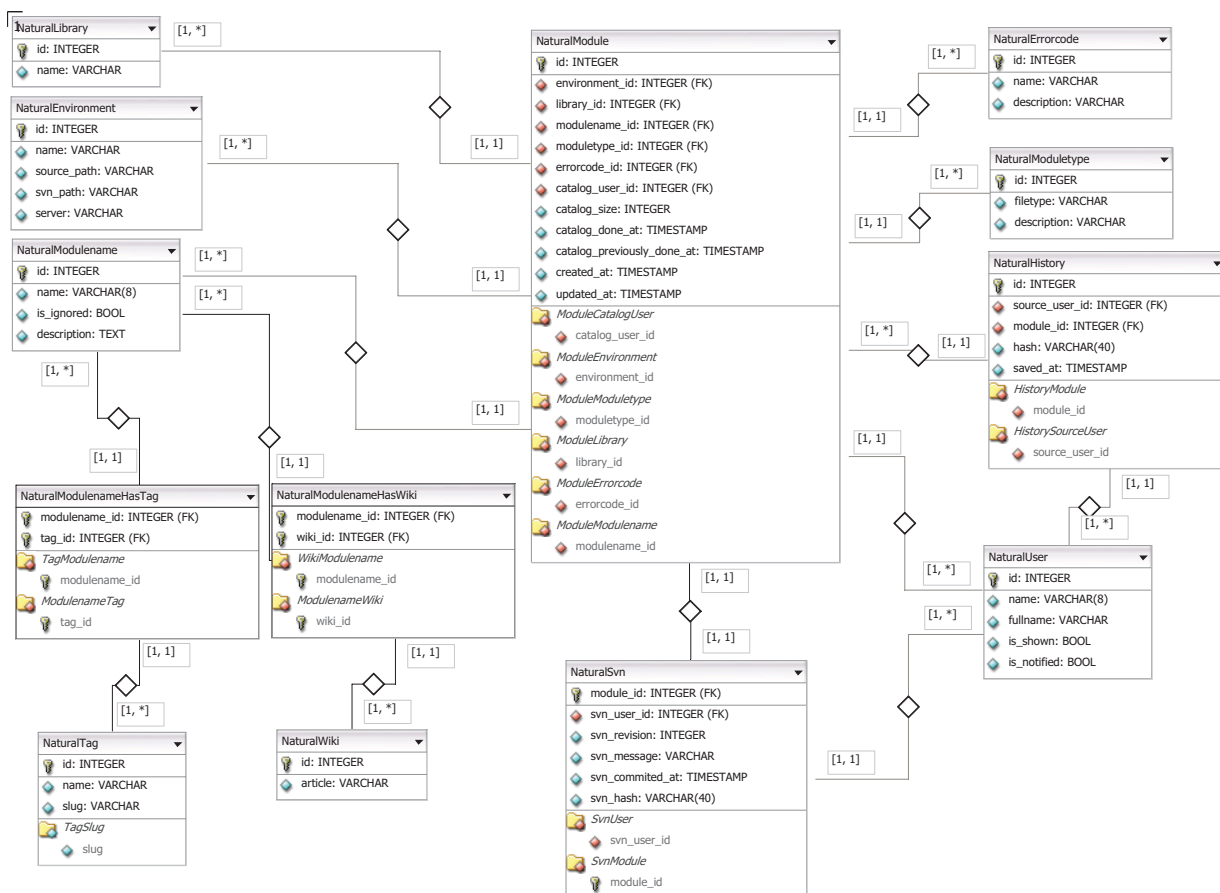
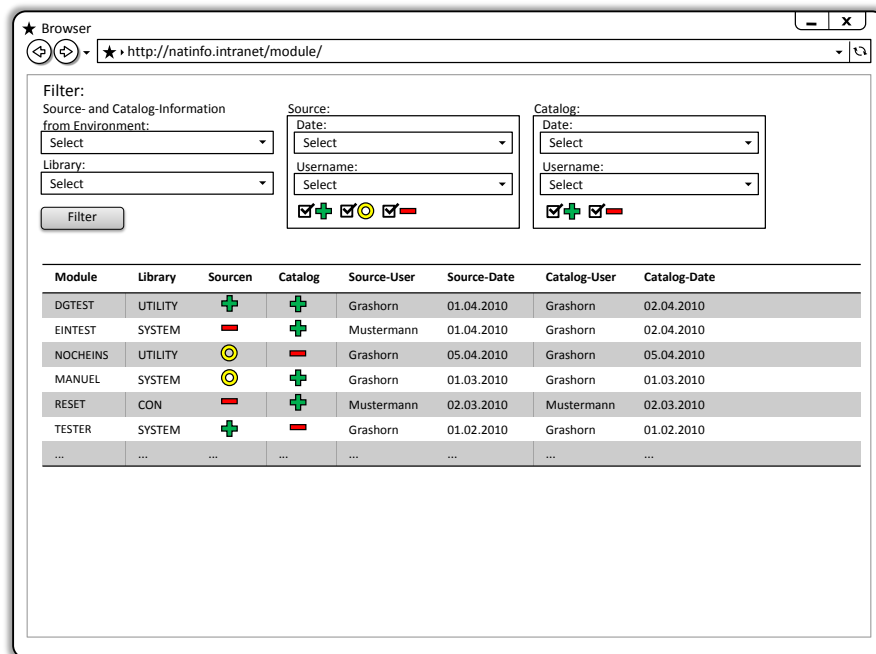


Abbildung 4: Datenbankmodell

A.6 Oberflächenentwürfe



The screenshot shows a web browser window with the address `http://natinfo.intranet/module/`. The interface includes a filter section at the top with the following fields:

- Filter:** Source- and Catalog-Information from Environment: (dropdown menu)
- Library:** (dropdown menu)
- Source:** Date: (dropdown menu), Username: (dropdown menu)
- Catalog:** Date: (dropdown menu), Username: (dropdown menu)
- Filter** button

Below the filter section is a table listing modules with their associated libraries, source and catalog status, and user information.

Module	Library	Sourcen	Catalog	Source-User	Source-Date	Catalog-User	Catalog-Date
DGTEST	UTILITY	+	+	Grashorn	01.04.2010	Grashorn	02.04.2010
EINTEST	SYSTEM	-	+	Mustermann	01.04.2010	Grashorn	02.04.2010
NOCHEINS	UTILITY	o	-	Grashorn	05.04.2010	Grashorn	05.04.2010
MANUEL	SYSTEM	o	+	Grashorn	01.03.2010	Grashorn	01.03.2010
RESET	CON	-	+	Mustermann	02.03.2010	Mustermann	02.03.2010
TESTER	SYSTEM	+	-	Grashorn	01.02.2010	Grashorn	01.02.2010
...

Abbildung 5: Liste der Module mit Filtermöglichkeiten

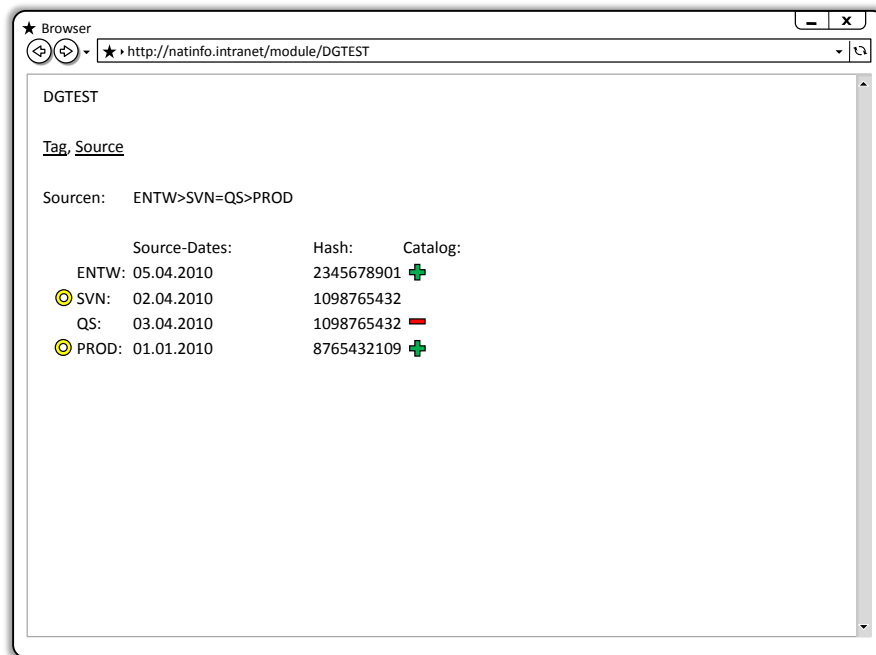


Abbildung 6: Anzeige der Übersichtsseite einzelner Module

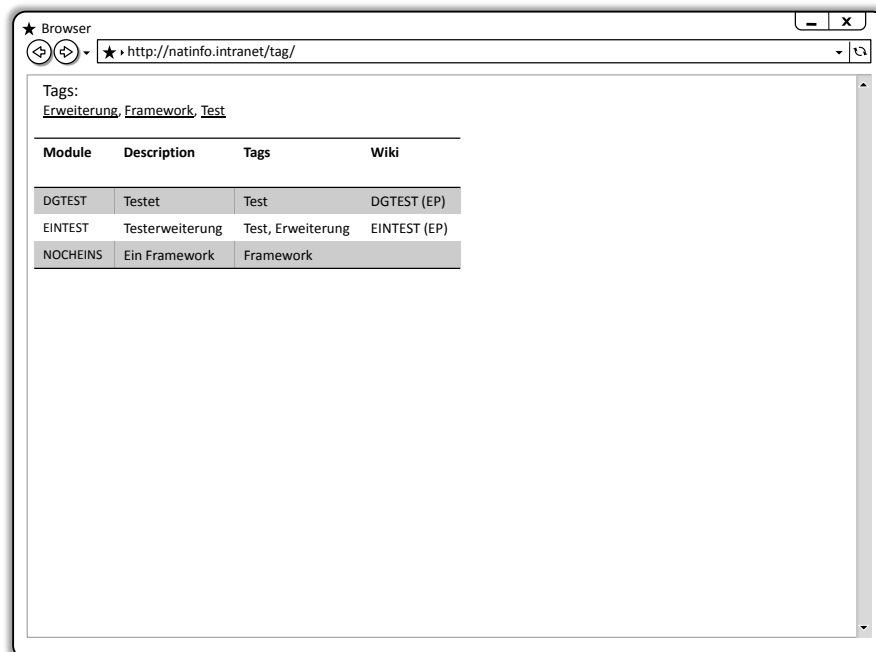


Abbildung 7: Anzeige und Filterung der Module nach Tags

A.7 Screenshots der Anwendung



Tags

Project, Test

Modulename	Description	Tags	Wiki
DGTEST	Macht einen ganz tollen Tab.	HGP	SMTAB_(EP), b
MALWAS		HGP, Test	
HDRGE		HGP, Project	
WURAM		HGP, Test	
PAMIU		HGP	

Abbildung 8: Anzeige und Filterung der Module nach Tags



Modules

Environment	ENTW
Library	Select
Catalog user	Select
Catalog date	Select
Source user	Select
Source date	Select
Reset Filter	











Name	Library	Source	Catalog	Source-User	Source-Date	Catalog-User	Catalog-Date
SMTAB	UTILITY			MACKE	01.04.2010 13:00	MACKE	01.04.2010 13:00
DGTAB	CON			GRASHORN	01.04.2010 13:00	GRASHORN	01.04.2010 13:00
DGTEST	SUP			GRASHORN	05.04.2010 13:00	GRASHORN	05.04.2010 13:00
OHNETAG	CON			GRASHORN	05.04.2010 13:00	GRASHORN	01.04.2010 15:12
OHNEWIKI	CON			GRASHORN	05.04.2010 13:00	MACKE	01.04.2010 15:12

Abbildung 9: Liste der Module mit Filtermöglichkeiten

A.8 Entwicklerdokumentation

lib-model

[class tree: lib-model] [index: lib-model] [all elements]

Packages:
lib-model

Files:
Naturalmodulename.php

Classes:
Naturalmodulename

Class: Naturalmodulename

Source Location: /Naturalmodulename.php

Class Overview

```
BaseNaturalmodulename
|
--Naturalmodulename
```

Subclass for representing a row from the 'NaturalModulename' table.

Methods

- [__construct](#)
- [getNaturalTags](#)
- [getNaturalWikis](#)
- [loadNaturalModuleInformation](#)
- [__toString](#)

Class Details

[line 10]
Subclass for representing a row from the 'NaturalModulename' table.

Adds some business logic to the base.

[\[Top \]](#)

Class Methods

constructor `__construct` [line 56]

```
Naturalmodulename __construct( )
```

Initializes internal state of Naturalmodulename object.

Tags:
see: parent::__construct()
access: public

[\[Top \]](#)

method `getNaturalTags` [line 68]

```
array getNaturalTags( )
```

Returns an Array of NaturalTags connected with this Modulename.

Tags:

return: Array of NaturalTags
access: public

[\[Top \]](#)

method getNaturalWikis [line 83]

```
array getNaturalWikis( )
```

Returns an Array of NaturalWikis connected with this Modulename.

Tags:

return: Array of NaturalWikis
access: public

[\[Top \]](#)

method loadNaturalModuleInformation [line 17]

```
ComparedNaturalModuleInformation  
loadNaturalModuleInformation( )
```

Gets the ComparedNaturalModuleInformation for this NaturalModulename.

Tags:

access: public

[\[Top \]](#)

method __toString [line 47]

```
string __toString( )
```

Returns the name of this NaturalModulename.

Tags:

access: public

[\[Top \]](#)

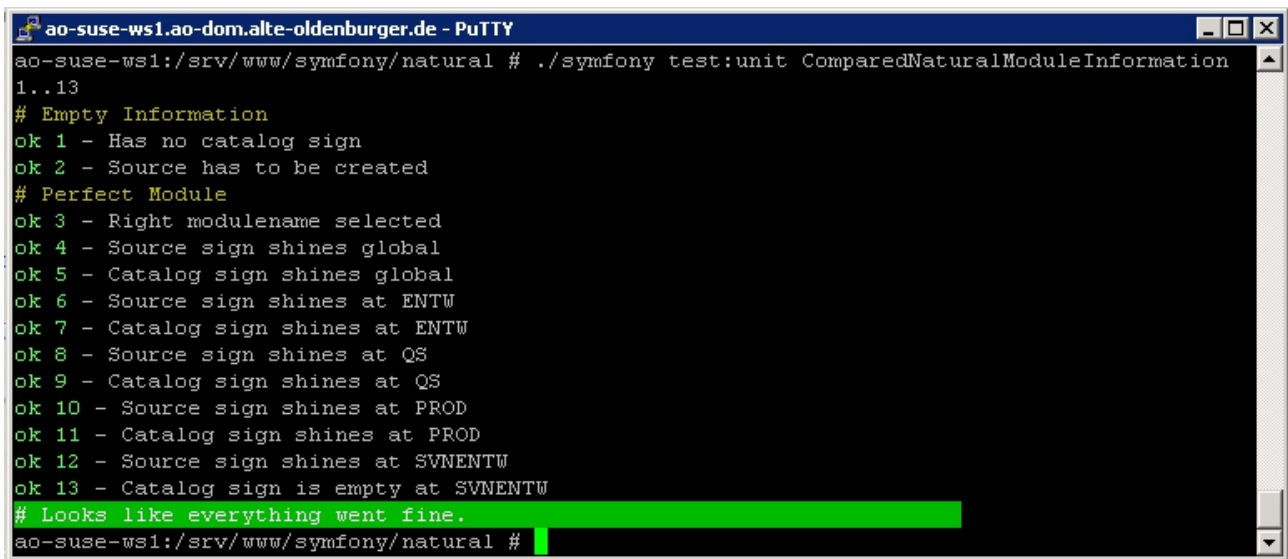
Documentation generated on Thu, 22 Apr 2010 08:14:01 +0200 by [phpDocumentor 1.4.2](#)

A.9 Testfall und sein Aufruf auf der Konsole

```
1 <?php
2 include(dirname(__FILE__).'/../bootstrap/Propel.php');
3
4 $t = new lime_test(13);
5
6 $t->comment('Empty Information');
7 $emptyComparedInformation = new ComparedNaturalModuleInformation(array());
8 $t->is($emptyComparedInformation->getCatalogSign(), ComparedNaturalModuleInformation::EMPTY_SIGN, '
    Has no catalog sign');
9 $t->is($emptyComparedInformation->getSourceSign(), ComparedNaturalModuleInformation::SIGN_CREATE, '
    Source has to be created');
10
11 $t->comment('Perfect Module');
12 $criteria = new Criteria();
13 $criteria->add(NaturalmodulenamePeer::NAME, 'SMTAB');
14 $moduleName = NaturalmodulenamePeer::doSelectOne($criteria);
15 $t->is($moduleName->getName(), 'SMTAB', 'Right modulename selected');
16 $comparedInformation = $moduleName->loadNaturalModuleInformation();
17 $t->is($comparedInformation->getSourceSign(), ComparedNaturalModuleInformation::SIGN_OK, 'Source sign
    shines global');
18 $t->is($comparedInformation->getCatalogSign(), ComparedNaturalModuleInformation::SIGN_OK, 'Catalog sign
    shines global');
19 $infos = $comparedInformation->getNaturalModuleInformations();
20 foreach($infos as $info)
21 {
22     $env = $info->getEnvironmentName();
23     $t->is($info->getSourceSign(), ComparedNaturalModuleInformation::SIGN_OK, 'Source sign shines at ' . $env);
24     if ($env != 'SVNENTW')
25     {
26         $t->is($info->getCatalogSign(), ComparedNaturalModuleInformation::SIGN_OK, 'Catalog sign shines at ' .
            $info->getEnvironmentName());
27     }
28     else
29     {
30         $t->is($info->getCatalogSign(), ComparedNaturalModuleInformation::EMPTY_SIGN, 'Catalog sign is empty
            at ' . $info->getEnvironmentName());
31     }
32 }
33 ?>
```

Listing 1: Testfall in PHP

A Anhang



```
ao-suse-ws1.ao-dom.alte-oldenburger.de - PuTTY
ao-suse-ws1:/srv/www/symfony/natural # ./symfony test:unit ComparedNaturalModuleInformation
1..13
# Empty Information
ok 1 - Has no catalog sign
ok 2 - Source has to be created
# Perfect Module
ok 3 - Right modulename selected
ok 4 - Source sign shines global
ok 5 - Catalog sign shines global
ok 6 - Source sign shines at ENTW
ok 7 - Catalog sign shines at ENTW
ok 8 - Source sign shines at QS
ok 9 - Catalog sign shines at QS
ok 10 - Source sign shines at PROD
ok 11 - Catalog sign shines at PROD
ok 12 - Source sign shines at SVNENTW
ok 13 - Catalog sign is empty at SVNENTW
# Looks like everything went fine.
ao-suse-ws1:/srv/www/symfony/natural #
```

Abbildung 10: Aufruf des Testfalls auf der Konsole

A.10 Klasse: ComparedNaturalModuleInformation

Kommentare und simple Getter/Setter werden nicht angezeigt.

```
1 <?php
2 class ComparedNaturalModuleInformation
3 {
4     const EMPTY_SIGN = 0;
5     const SIGN_OK = 1;
6     const SIGN_NEXT_STEP = 2;
7     const SIGN_CREATE = 3;
8     const SIGN_CREATE_AND_NEXT_STEP = 4;
9     const SIGN_ERROR = 5;
10
11     private $naturalModuleInformations = array();
12
13     public static function environments()
14     {
15         return array("ENTW", "SVNENTW", "QS", "PROD");
16     }
17
18     public static function signOrder()
19     {
20         return array(self::SIGN_ERROR, self::SIGN_NEXT_STEP, self::SIGN_CREATE_AND_NEXT_STEP, self::SIGN_CREATE, self::SIGN_OK);
21     }
22
23     public function __construct(array $naturalInformations)
24     {
25         $this->allocateModulesToEnvironments($naturalInformations);
```

A Anhang

```
26     $this->allocateEmptyModulesToMissingEnvironments();
27     $this->determineSourceSignsForAllEnvironments();
28 }
29
30 private function allocateModulesToEnvironments(array $naturalInformations)
31 {
32     foreach ($naturalInformations as $naturalInformation)
33     {
34         $env = $naturalInformation->getEnvironmentName();
35         if (in_array($env, self::environments()))
36         {
37             $this->naturalModuleInformations[array_search($env, self::environments())] = $naturalInformation;
38         }
39     }
40 }
41
42 private function allocateEmptyModulesToMissingEnvironments()
43 {
44     if (array_key_exists(0, $this->naturalModuleInformations))
45     {
46         $this->naturalModuleInformations[0]->setSourceSign(self::SIGN_OK);
47     }
48
49     for ($i = 0; $i < count(self::environments()); $i++)
50     {
51         if (!array_key_exists($i, $this->naturalModuleInformations))
52         {
53             $environments = self::environments();
54             $this->naturalModuleInformations[$i] = new EmptyNaturalModuleInformation($environments[$i]);
55             $this->naturalModuleInformations[$i]->setSourceSign(self::SIGN_CREATE);
56         }
57     }
58 }
59
60 public function determineSourceSignsForAllEnvironments()
61 {
62     for ($i = 1; $i < count(self::environments()); $i++)
63     {
64         $currentInformation = $this->naturalModuleInformations[$i];
65         $previousInformation = $this->naturalModuleInformations[$i - 1];
66         if ($currentInformation->getSourceSign() <> self::SIGN_CREATE)
67         {
68             if ($previousInformation->getSourceSign() <> self::SIGN_CREATE)
69             {
70                 if ($currentInformation->getHash() <> $previousInformation->getHash())
71                 {
72                     if ($currentInformation->getSourceDate('YmdHis') > $previousInformation->getSourceDate('YmdHis'))
73                     {
74                         $currentInformation->setSourceSign(self::SIGN_ERROR);
75                     }
76                 }
77             }
78         }
79     }
80 }
```

A Anhang

```
76         else
77         {
78             $currentInformation->setSourceSign(self::SIGN_NEXT_STEP);
79         }
80     }
81     else
82     {
83         $currentInformation->setSourceSign(self::SIGN_OK);
84     }
85 }
86 else
87 {
88     $currentInformation->setSourceSign(self::SIGN_ERROR);
89 }
90 }
91 elseif ($previousInformation->getSourceSign() <> self::SIGN_CREATE && $previousInformation->
    getSourceSign() <> self::SIGN_CREATE_AND_NEXT_STEP)
92 {
93     $currentInformation->setSourceSign(self::SIGN_CREATE_AND_NEXT_STEP);
94 }
95 }
96 }
97
98 private function containsSourceSign($sign)
99 {
100     foreach($this->naturalModuleInformations as $information)
101     {
102         if($information->getSourceSign() == $sign)
103         {
104             return true;
105         }
106     }
107     return false ;
108 }
109
110 private function containsCatalogSign($sign)
111 {
112     foreach($this->naturalModuleInformations as $information)
113     {
114         if($information->getCatalogSign() == $sign)
115         {
116             return true;
117         }
118     }
119     return false ;
120 }
121 }
122 ?>
```

Listing 2: Klasse: ComparedNaturalModuleInformation

A.11 Klassendiagramm

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

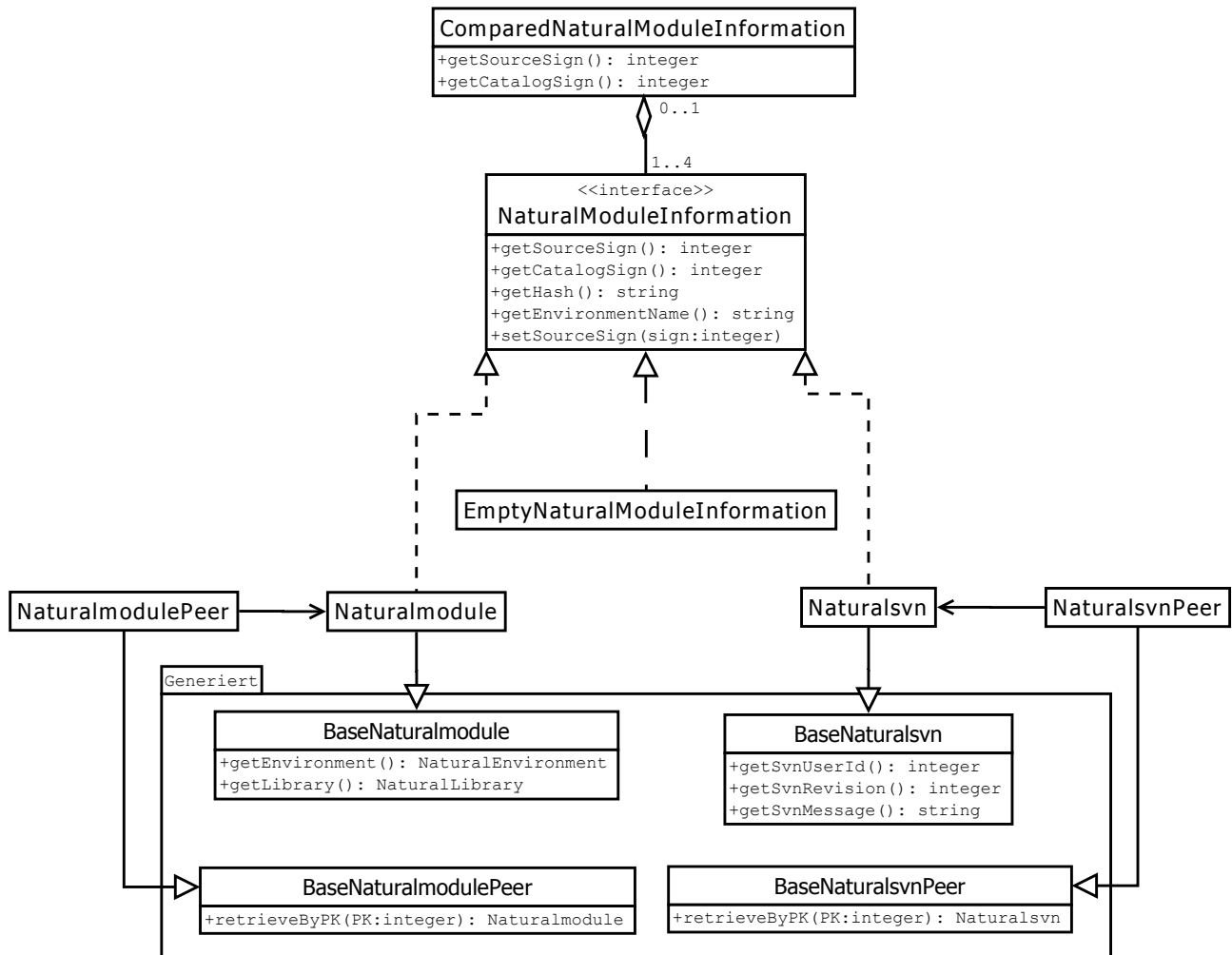







Abbildung 11: Klassendiagramm

A.12 Benutzerdokumentation

Ausschnitt aus der Benutzerdokumentation:

Symbol	Bedeutung global	Bedeutung einzeln
	Alle Module weisen den gleichen Stand auf.	Das Modul ist auf dem gleichen Stand wie das Modul auf der vorherigen Umgebung.
	Es existieren keine Module (fachlich nicht möglich).	Weder auf der aktuellen noch auf der vorherigen Umgebung sind Module angelegt. Es kann also auch nichts übertragen werden.
	Ein Modul muss durch das Übertragen von der vorherigen Umgebung erstellt werden.	Das Modul der vorherigen Umgebung kann übertragen werden, auf dieser Umgebung ist noch kein Modul vorhanden.
	Auf einer vorherigen Umgebung gibt es ein Modul, welches übertragen werden kann, um das nächste zu aktualisieren.	Das Modul der vorherigen Umgebung kann übertragen werden um dieses zu aktualisieren.
	Ein Modul auf einer Umgebung wurde entgegen des Entwicklungsprozesses gespeichert.	Das aktuelle Modul ist neuer als das Modul auf der vorherigen Umgebung oder die vorherige Umgebung wurde übersprungen.