

## Über diese Vorlage

Diese  $\text{\LaTeX}$ -Vorlage wurde von Stefan Macke<sup>1</sup> als Grundlage für die Projektdokumentationen der Auszubildenden zum Fachinformatiker mit Fachrichtung Anwendungsentwicklung bei der ALTE OLDENBURGER Krankenversicherung entwickelt. Nichtsdestotrotz dürfte sie ebenso für die anderen IT-Berufe<sup>2</sup> geeignet sein, da diese anhand der gleichen Verordnung bewertet werden.

Diese Vorlage enthält bereits eine Vorstrukturierung der möglichen Inhalte einer tatsächlichen Projektdokumentation, die auf Basis der Erfahrungen im Rahmen der Prüfertätigkeit des Autors erstellt und unter Zuhilfenahme von ROHRER UND SEDLACEK [2011] abgerundet wurden.

Sämtliche verwendeten Abbildungen, Tabellen und Listings stammen von GRASHORN [2010].

Download-Link für diese Vorlage: <http://fiae.link/LaTeXVorlageFIAE>

Auch verfügbar auf GitHub: <https://github.com/StefanMacke/latex-vorlage-fiae>

## Lizenz



Dieses Werk steht unter einer Creative Commons Namensnennung - Weitergabe unter gleichen Bedingungen 4.0 International Lizenz.<sup>3</sup>



**Namensnennung** Sie müssen den Namen des Autors/Rechteinhabers in der von ihm festgelegten Weise nennen.<sup>4</sup>

**Weitergabe unter gleichen Bedingungen** Wenn Sie das lizenzierte Werk bzw. den lizenzierten Inhalt bearbeiten oder in anderer Weise erkennbar als Grundlage für eigenes Schaffen verwenden, dürfen Sie die daraufhin neu entstandenen Werke bzw. Inhalte nur unter Verwendung von Lizenzbedingungen weitergeben, die mit denen dieses Lizenzvertrages identisch oder vergleichbar sind.

---

<sup>1</sup>Blog des Autors: <http://fachinformatiker-anwendungsentwicklung.net>, Twitter: @StefanMacke

<sup>2</sup>z. B. IT-Kaufleute, Fachinformatiker mit Fachrichtung Systemintegration usw.

<sup>3</sup><http://creativecommons.org/licenses/by-sa/4.0/>

<sup>4</sup>Die Namensnennung im  $\text{\LaTeX}$ -Quelltext mit Link auf <http://fiae.link/LaTeXVorlageFIAE> reicht hierfür aus.

## Inhalt der Projektdokumentation

Grundsätzlich definiert die [REGIERUNG DER BUNDESREPUBLIK DEUTSCHLAND](#) [1997, S. 1746]<sup>5</sup> das Ziel der Projektdokumentation wie folgt:

„Durch die Projektarbeit und deren Dokumentation soll der Prüfling belegen, daß er Arbeitsabläufe und Teilaufgaben zielorientiert unter Beachtung wirtschaftlicher, technischer, organisatorischer und zeitlicher Vorgaben selbständig planen und kundengerecht umsetzen sowie Dokumentationen kundengerecht anfertigen, zusammenstellen und modifizieren kann.“

Und das [BUNDESMINISTERIUM FÜR BILDUNG UND FORSCHUNG](#) [2000, S. 36] ergänzt:

„Die Ausführung der Projektarbeit wird mit praxisbezogenen Unterlagen dokumentiert. Der Prüfungsausschuss bewertet die Projektarbeit anhand der Dokumentation. Dabei wird nicht das Ergebnis – z. B. ein lauffähiges Programm – herangezogen, sondern der Arbeitsprozess. Die Dokumentation ist keine wissenschaftliche Abhandlung, sondern eine handlungsorientierte Darstellung des Projektablaufs mit praxisbezogenen, d.h. betriebüblichen Unterlagen. Sie soll einen Umfang von maximal 10 bis 15 DIN A 4-Seiten nicht überschreiten. Soweit erforderlich können in einem Anhang z. B. den Zusammenhang erläuternde Darstellungen beigelegt werden.“

Außerdem werden dort die grundlegenden Inhalte der Projektdokumentation aufgelistet:

- Name und Ausbildungsberuf des Prüfungsteilnehmers
- Angabe des Ausbildungsbetriebes
- Thema der Projektarbeit
- Falls erforderlich, Beschreibung/Konkretisierung des Auftrages
- Umfassende Beschreibung der Prozessschritte und der erzielten Ergebnisse
- Gegebenenfalls Veränderungen zum Projektantrag mit Begründung
- Wenn für das Projekt erforderlich, ein Anhang mit praxisbezogenen Unterlagen und Dokumenten. Dieser Anhang sollte nicht aufgebläht werden. Die angehängten Dokumente und Unterlagen sind auf das absolute Minimum zu beschränken.

In den folgenden Kapiteln werden diese geforderten Inhalte und sinnvolle Ergänzungen nun meist stichwortartig und ggfs. mit Beispielen beschrieben. Nicht alle Kapitel müssen in jeder Dokumentation vorhanden sein. Handelt es sich bspw. um ein in sich geschlossenes Projekt, kann das Kapitel [1.5: Projektabgrenzung](#) entfallen; arbeitet die Anwendung nur mit [XML](#)-Dateien, kann und muss keine Datenbank beschrieben werden usw.

---

<sup>5</sup>Dieses Dokument sowie alle weiteren hier genannten können unter <http://fiae.link/LaTeXVorlageFIAEQuellen> heruntergeladen werden.

## Formale Vorgaben

Die formalen Vorgaben zum Umfang und zur Gestaltung der Projektdokumentation können je nach IHK recht unterschiedlich sein. Normalerweise sollte die zuständige IHK einen Leitfaden bereitstellen, in dem alle Formalien nachgelesen werden können, wie z. B. bei der [IHK OLDENBURG \[2006\]](#).

Als Richtwert verwende ich 15 Seiten für den reinen Inhalt. Also in dieser Vorlage alle Seiten, die arabisch nummeriert sind (ohne das Literaturverzeichnis und die eidesstattliche Erklärung). Große Abbildungen, Quelltexte, Tabellen usw. gehören in den Anhang, der 25 Seiten nicht überschreiten sollte.

Typographische Konventionen, Seitenränder usw. können in der Datei `Seitenstil.tex` beliebig angepasst werden.

## Bewertungskriterien

Die Bewertungskriterien für die Benotung der Projektdokumentation sind recht einheitlich und können leicht in Erfahrung gebracht werden, z. B. bei der [IHK DARMSTADT \[2011\]](#). Grundsätzlich sollte die Projektdokumentation nach der Fertigstellung noch einmal im Hinblick auf diese Kriterien durchgeschaut werden.



Abschlussprüfung Sommer 2015

Fachinformatiker für Anwendungsentwicklung  
Dokumentation zur betrieblichen Projektarbeit

## Entwicklung von NatInfo

**Webbasiertes Tool zur Unterstützung der Entwickler**

Abgabetermin: Vechta, den 23.04.2015

**Prüfungsbewerber:**

Stefan Macke  
Meine Straße 55  
12345 Musterstadt



**Ausbildungsbetrieb:**

ALTE OLDENBURGER Krankenversicherung AG  
Theodor-Heuss-Str. 96  
49377 Vechta

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

<b>Abbildungsverzeichnis</b>	<b>II</b>
<b>Tabellenverzeichnis</b>	<b>1</b>
<b>Listings</b>	<b>1</b>
<b>Abkürzungsverzeichnis</b>	<b>1</b>
<b>1 Einleitung</b>	<b>2</b>
1.1 Projektumfeld . . . . .	2
1.2 Projektziel . . . . .	2
1.3 Projektbegründung . . . . .	2
1.4 Projektschnittstellen . . . . .	2
1.5 Projektabgrenzung . . . . .	2
<b>2 Projektplanung</b>	<b>3</b>
2.1 Projektphasen . . . . .	3
2.2 Abweichungen vom Projektantrag . . . . .	3
2.3 Ressourcenplanung . . . . .	3
2.4 Entwicklungsprozess . . . . .	3
<b>3 Analysephase</b>	<b>4</b>
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 . . . . .	5
3.4 Anwendungsfälle . . . . .	5
3.5 Qualitätsanforderungen . . . . .	6
3.6 Lastenheft/Fachkonzept . . . . .	6
<b>4 Entwurfsphase</b>	<b>6</b>
4.1 Zielplattform . . . . .	6
4.2 Architekturdesign . . . . .	6
4.3 Entwurf der Benutzeroberfläche . . . . .	7
4.4 Datenmodell . . . . .	7
4.5 Geschäftslogik . . . . .	8
4.6 Maßnahmen zur Qualitätssicherung . . . . .	8
4.7 Pflichtenheft/Datenverarbeitungskonzept . . . . .	8

## Abbildungsverzeichnis

<b>5</b>	<b>Implementierungsphase</b>	<b>9</b>
5.1	Implementierung der Datenstrukturen . . . . .	9
5.2	Implementierung der Benutzeroberfläche . . . . .	9
5.3	Implementierung der Geschäftslogik . . . . .	9
<b>6</b>	<b>Abnahmephase</b>	<b>9</b>
<b>7</b>	<b>Einführungsphase</b>	<b>10</b>
<b>8</b>	<b>Dokumentation</b>	<b>10</b>
<b>9</b>	<b>Fazit</b>	<b>10</b>
9.1	Soll-/Ist-Vergleich . . . . .	10
9.2	Lessons Learned . . . . .	11
9.3	Ausblick . . . . .	11
	<b>Literaturverzeichnis</b>	<b>11</b>
	<b>Eidesstattliche Erklärung</b>	<b>13</b>
<b>A</b>	<b>Anhang</b>	<b>i</b>
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
A.10	Klasse: ComparedNaturalModuleInformation . . . . .	xiii
A.11	Klassendiagramm . . . . .	xvi
A.12	Benutzerdokumentation . . . . .	xvii

## Abbildungsverzeichnis

1	Vereinfachtes ER-Modell . . . . .	7
2	Prozess des Einlesens eines Moduls . . . . .	8
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	Zeitplanung . . . . .	3
2	Kostenaufstellung . . . . .	5
3	Entscheidungsmatrix . . . . .	7
4	Soll-/Ist-Vergleich . . . . .	11

## Listings

1	Testfall in PHP . . . . .	xii
2	Klasse: ComparedNaturalModuleInformation . . . . .	xiii

## Abkürzungsverzeichnis

<b>API</b>	Application Programming Interface . . . . .	10
<b>CSV</b>	Comma Separated Value . . . . .	iv
<b>EPK</b>	Ereignisgesteuerte Prozesskette . . . . .	5
<b>ERM</b>	Entity-Relationship-Modell . . . . .	7
<b>HTML</b>	Hypertext Markup Language . . . . .	9
<b>MVC</b>	Model View Controller . . . . .	6
<b>NatInfo</b>	Natural Information System . . . . .	iv
<b>Natural</b>	Programmiersprache der Software AG . . . . .	v
<b>PHP</b>	Hypertext Preprocessor . . . . .	6
<b>SQL</b>	Structured Query Language . . . . .	9
<b>SVN</b>	Subversion . . . . .	iv
<b>UML</b>	Unified Modeling Language . . . . .	iii
<b>XML</b>	Extensible Markup Language . . . . .	9

## 1 Einleitung

### 1.1 Projektumfeld

Der Praktikumsbetrieb beschäftigt 20 Mitarbeiter, davon sollen 15 Personen das Ergebnis dieses Projektes nutzen können.

Falls ein anderes Projekt wichtig für dieses Abschlussprojekt ist, soll das deutlich abgegrenzt werden (bzw. schon im Antrag, siehe auch Projektabgrenzung [1.5](#)).

Zum Umfeld gehören auch die Schnittstellen, s. [1.4](#).

### 1.2 Projektziel

- Worum geht es eigentlich?
- Was soll erreicht werden?

### 1.3 Projektbegründung

- Warum ist das Projekt sinnvoll (z. B. Kosten- oder Zeitersparnis, weniger Fehler)?
- Was ist die Motivation hinter dem Projekt?

### 1.4 Projektschnittstellen

- Mit welchen anderen Systemen interagiert die Anwendung (technische Schnittstellen)?
- Wer genehmigt das Projekt bzw. stellt Mittel zur Verfügung?
- Wer sind die Benutzer der Anwendung?
- Wem muss das Ergebnis präsentiert werden?

### 1.5 Projektabgrenzung

- Was ist explizit nicht Teil des Projekts (insb. bei Teilprojekten)?



## 2 Projektplanung

### 2.1 Projektphasen

Beginn war der 10. Januar 2022 und Ende im März 2022.

- Verfeinerung der Zeitplanung, die bereits im Projektantrag vorgestellt wurde.

**Beispiel** Tabelle 1 zeigt ein Beispiel für eine grobe Zeitplanung.

Projektphase	Geplante Zeit
Analysephase	9 h
Entwurfsphase	19 h
Implementierungsphase	29 h
Abnahmetest der Fachabteilung	1 h
Einführungsphase	1 h
Erstellen der Dokumentation	9 h
Pufferzeit	2 h
<b>Gesamt</b>	<b>70 h</b>

Tabelle 1: Zeitplanung

Eine detailliertere Zeitplanung findet sich im Anhang [A.1: Detaillierte Zeitplanung](#) auf Seite i.

### 2.2 Abweichungen vom Projektantrag

- Sollte es Abweichungen zum Projektantrag geben (z. B. Zeitplanung, Inhalt des Projekts, neue Anforderungen), müssen diese explizit aufgeführt und begründet werden.

### 2.3 Ressourcenplanung

- Detaillierte Planung der benötigten Ressourcen (Hard-/Software, Räumlichkeiten usw.).
- Ggfs. sind auch personelle Ressourcen einzuplanen (z. B. unterstützende Mitarbeiter).
- Hinweis: Häufig werden hier Ressourcen vergessen, die als selbstverständlich angesehen werden (z. B. PC, Büro).

### 2.4 Entwicklungsprozess

- Welcher Entwicklungsprozess wird bei der Bearbeitung des Projekts verfolgt (z. B. Wasserfall, agiler Prozess)?

## 3 Analysephase

### 3.1 Ist-Analyse

- Wie ist die bisherige Situation (z. B. bestehende Programme, Wünsche der Mitarbeiter)?
- Was gilt es zu erstellen/verbessern?

### 3.2 Wirtschaftlichkeitsanalyse

- Lohnt sich das Projekt für das Unternehmen?

#### 3.2.1 „Make or Buy“-Entscheidung

- Gibt es vielleicht schon ein fertiges Produkt, dass alle Anforderungen des Projekts abdeckt?
- Wenn ja, wieso wird das Projekt trotzdem umgesetzt?

#### 3.2.2 Projektkosten

- Welche Kosten fallen bei der Umsetzung des Projekts im Detail an (z. B. Entwicklung, Einführung/-Schulung, Wartung)?

**Beispielrechnung (verkürzt)** Die Kosten für die Durchführung des Projekts setzen sich sowohl aus Personal-, als auch aus Ressourcenkosten zusammen. Laut Tarifvertrag verdient ein Auszubildender im dritten Lehrjahr pro Monat 1000 € Brutto.

$$8 \text{ h/Tag} \cdot 220 \text{ Tage/Jahr} = 1760 \text{ h/Jahr} \quad (1)$$

$$1000 \text{ €/Monat} \cdot 13,3 \text{ Monate/Jahr} = 13300 \text{ €/Jahr} \quad (2)$$

$$\frac{13300 \text{ €/Jahr}}{1760 \text{ h/Jahr}} \approx 7,56 \text{ €/h} \quad (3)$$

Es ergibt sich also ein Stundenlohn von 7,56 €. Die Durchführungszeit des Projekts beträgt 70 Stunden. Für die Nutzung von Ressourcen<sup>6</sup> wird ein pauschaler Stundensatz von 15 € angenommen. Für die anderen Mitarbeiter wird pauschal ein Stundenlohn von 25 € angenommen. Eine Aufstellung der Kosten befindet sich in Tabelle 2 und sie betragen insgesamt 2739,20 €.

---

<sup>6</sup>Räumlichkeiten, Arbeitsplatzrechner etc.

### 3 Analysephase

Vorgang	Zeit	Kosten pro Stunde	Kosten
Entwicklungskosten	70 h	7,56 € + 15 € = 22,56 €	1579,20 €
Fachgespräch	3 h	25 € + 15 € = 40 €	120 €
Abnahmetest	1 h	25 € + 15 € = 40 €	40 €
Anwenderschulung	25 h	25 € + 15 € = 40 €	1000 €
			<b>2739,20 €</b>

Tabelle 2: Kostenaufstellung

#### 3.2.3 Amortisationsdauer

- Welche monetären Vorteile bietet das Projekt (z. B. Einsparung von Lizenzkosten, Arbeitszeiterparnis, bessere Usability, Korrektheit)?
- Wann hat sich das Projekt amortisiert?

**Beispielrechnung (verkürzt)** Bei einer Zeiteinsparung von 10 Minuten am Tag für jeden der 25 Anwender und 220 Arbeitstagen im Jahr ergibt sich eine gesamte Zeiteinsparung von

$$25 \cdot 220 \text{ Tage/Jahr} \cdot 10 \text{ min/Tag} = 55000 \text{ min/Jahr} \approx 917 \text{ h/Jahr} \quad (4)$$

Dadurch ergibt sich eine jährliche Einsparung von

$$917 \text{ h} \cdot (25 + 15) \text{ €/h} = 36680 \text{ €} \quad (5)$$

Die Amortisationszeit beträgt also  $\frac{2739,20 \text{ €}}{36680 \text{ €/Jahr}} \approx 0,07 \text{ Jahre} \approx 4 \text{ Wochen}$ .

### 3.3 Nutzwertanalyse

- Darstellung des nicht-monetären Nutzens (z. B. Vorher-/Nachher-Vergleich anhand eines Wirtschaftlichkeitskoeffizienten).

**Beispiel** Ein Beispiel für eine Entscheidungsmatrix findet sich in Kapitel 4.2: [Architekturdesign](#).

### 3.4 Anwendungsfälle

- Welche Anwendungsfälle soll das Projekt abdecken?
- Einer oder mehrere interessante (!) Anwendungsfälle könnten exemplarisch durch ein Aktivitätsdiagramm oder eine Ereignisgesteuerte Prozesskette (EPK) detailliert beschrieben werden.

## 4 Entwurfsphase

---

**Beispiel** Ein Beispiel für ein Use Case-Diagramm findet sich im Anhang [A.3: Use Case-Diagramm](#) auf Seite [iii](#).

### 3.5 Qualitätsanforderungen

- Welche Qualitätsanforderungen werden an die Anwendung gestellt (z. B. hinsichtlich Performance, Usability, Effizienz etc. (siehe [ISO/IEC 9126-1 \[2001\]](#)))?

### 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](#).

## 4 Entwurfsphase

### 4.1 Zielplattform

- Beschreibung der Kriterien zur Auswahl der Zielplattform (u. a. Programmiersprache, Datenbank, Client/Server, Hardware).

### 4.2 Architekturdesign

- Beschreibung und Begründung der gewählten Anwendungsarchitektur (z. B. Model View Controller ([MVC](#))).
- Ggfs. Bewertung und Auswahl von verwendeten Frameworks sowie ggfs. eine kurze Einführung in die Funktionsweise des verwendeten Frameworks.

**Beispiel** Anhand der Entscheidungsmatrix in Tabelle [3](#) wurde für die Implementierung der Anwendung das Hypertext Preprocessor ([PHP](#))-Framework [Symfony<sup>7</sup>](#) ausgewählt.

---

<sup>7</sup>Vgl. [SENSIO LABS \[2010\]](#).

Eigenschaft	Gewichtung	Akelos	CakePHP	Symfony	Eigenentwicklung
Dokumentation	5	4	3	5	0
Reenginierung	3	4	2	5	3
Generierung	3	5	5	5	2
Testfälle	2	3	2	3	3
Standardaufgaben	4	3	3	3	0
<b>Gesamt:</b>	<b>17</b>	<b>65</b>	<b>52</b>	<b>73</b>	<b>21</b>
<b>Nutzwert:</b>		<b>3,82</b>	<b>3,06</b>	<b>4,29</b>	<b>1,24</b>

Tabelle 3: Entscheidungsmatrix

### 4.3 Entwurf der Benutzeroberfläche

- Entscheidung für die gewählte Benutzeroberfläche (z. B. GUI, Webinterface).
- Beschreibung des visuellen Entwurfs der konkreten Oberfläche (z. B. Mockups, Menüführung).
- Ggfs. Erläuterung von angewendeten Richtlinien zur Usability und Verweis auf Corporate Design.

**Beispiel** Beispielentwürfe finden sich im Anhang [A.6: Oberflächenentwürfe](#) auf Seite [vi](#).

### 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.

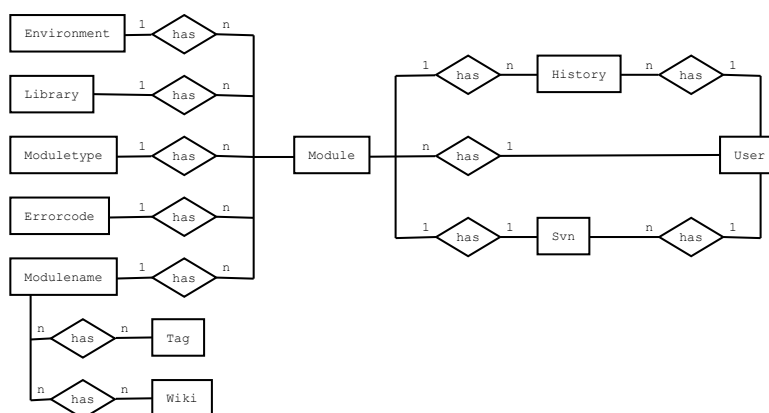


Abbildung 1: Vereinfachtes ER-Modell

## 4.5 Geschäftslogik

- Modellierung und Beschreibung der wichtigsten (!) Bereiche der Geschäftslogik (z. B. mit Komponenten-, Klassen-, Sequenz-, Datenflussdiagramm, Programmablaufplan, Struktogramm, EPK).
- 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.

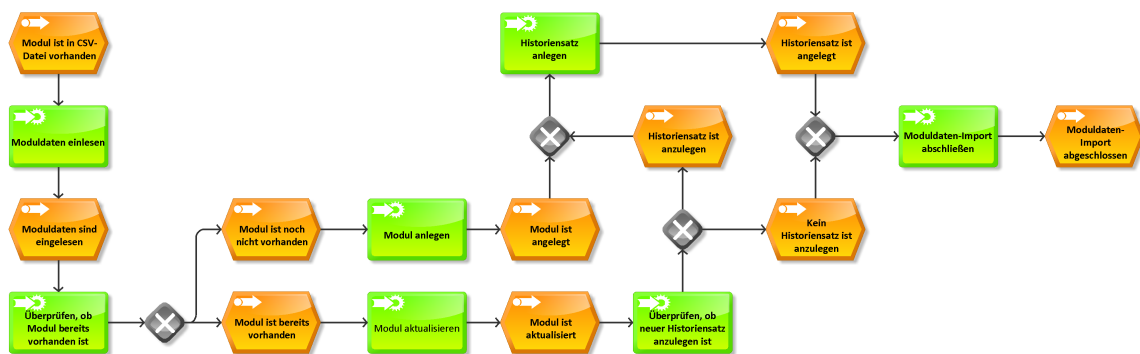


Abbildung 2: Prozess des Einlesens eines Moduls

## 4.6 Maßnahmen zur Qualitätssicherung

- Welche Maßnahmen werden ergriffen, um die Qualität des Projektergebnisses (siehe Kapitel [3.5: Qualitätsanforderungen](#)) zu sichern (z. B. automatische Tests, Anwendertests)?
- Ggfs. Definition von Testfällen und deren Durchführung (durch Programme/Benutzer).

## 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.

## 5 Implementierungsphase

### 5.1 Implementierung der Datenstrukturen

- Beschreibung der angelegten Datenbank (z. B. Generierung von Structured Query Language ([SQL](#)) aus Modellierungswerkzeug oder händisches Anlegen), Extensible Markup Language ([XML](#))-Schemas, usw..

### 5.2 Implementierung der Benutzeroberfläche

- Beschreibung der Implementierung der Benutzeroberfläche, falls dies separat zur Implementierung der Geschäftslogik erfolgt (z. B. bei Hypertext Markup Language ([HTML](#))-Oberflächen und Stylesheets).
- Ggfs. Beschreibung des Corporate Designs und dessen Umsetzung in der Anwendung.
- Screenshots der Anwendung

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

### 5.3 Implementierung der Geschäftslogik

- Beschreibung des Vorgehens bei der Umsetzung/Programmierung der entworfenen Anwendung.
- Ggfs. interessante Funktionen/Algorithmen im Detail vorstellen, verwendete Entwurfsmuster zeigen.
- Quelltextbeispiele zeigen.
- Hinweis: Wie in Kapitel [1: Einleitung](#) zitiert, wird nicht ein lauffähiges Programm bewertet, sondern die Projektdurchführung. Dennoch würde ich immer Quelltextausschnitte zeigen, da sonst Zweifel an der tatsächlichen Leistung des Prüflings aufkommen können.

**Beispiel** Die Klasse `ComparedNaturalModuleInformation` findet sich im Anhang [A.10: Klasse: ComparedNaturalModuleInformation](#) auf Seite [xiii](#).

## 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.

## 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?

## 8 Dokumentation

- Wie wurde die Anwendung für die Benutzer/Administratoren/Entwickler dokumentiert (z. B. Benutzerhandbuch, Application Programming Interface (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 PHPDoc<sup>8</sup> automatisch generiert. Ein beispielhafter Auszug aus der Dokumentation einer Klasse findet sich im Anhang [A.8: Entwicklerdokumentation](#) auf Seite [x](#).

## 9 Fazit

### 9.1 Soll-/Ist-Vergleich

- Projektziel erreicht?
- 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?

---

<sup>8</sup>Vgl. [PHPDOC.ORG](http://PHPDOC.ORG) [2010]



- 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 4 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
<b>Gesamt</b>	<b>70 h</b>	<b>70 h</b>	

Tabelle 4: 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

- [Bundesministerium für Bildung und Forschung 2000] BUNDESMINISTERIUM FÜR BILDUNG UND FORSCHUNG: Umsetzungshilfen für die neue Prüfungsstruktur der IT-Berufe / Bundesministerium für Bildung und Forschung. Version: Juli 2000. <http://fiaelink/UmsetzungshilfenITBerufe>. Bonn : ., Juli 2000 (1). – Abschlussbericht. – 476 S. – Notiz1
- [Grashorn 2010] GRASHORN, Dirk: Entwicklung von NatInfo – Webbasiertes Tool zur Unterstützung der Entwickler / Alte Oldenburger Krankenversicherung AG. Vechta : Springer-Verlag, April 2010 (2). – Dokumentation zur Projektarbeit. – Notiz2

- [IHK Darmstadt 2011] IHK DARMSTADT: *Bewertungsmatrix für Fachinformatiker/innen Anwendungsentwicklung*. <http://fiae.link/BewertungsmatrixDokuDarmstadt>. Version: März 2011
- [IHK Oldenburg 2006] IHK OLDENBURG: *Merkblatt zur Abschlussprüfung der IT-Berufe*. <http://fiae.link/MerkblattDokuOldenburg>. Version: Mai 2006
- [ISO/IEC 9126-1 2001] ISO/IEC 9126-1: *Software-Engineering – Qualität von Software-Produkten – Teil 1: Qualitätsmodell*. Juni 2001
- [phpdoc.org 2010] PHPDOC.ORG: *phpDocumentor-Website*. Version: 2010. <http://www.phpdoc.org/>, Abruf: 20.04.2010
- [Regierung der Bundesrepublik Deutschland 1997] REGIERUNG DER BUNDESREPUBLIK DEUTSCHLAND: *Verordnung über die Berufsausbildung im Bereich der Informations- und Telekommunikationstechnik*. <http://fiae.link/VerordnungITBerufe>. Version: Juli 1997
- [Rohrer und Sedlacek 2011] ROHRER, Anselm ; SEDLACEK, Ramona: *Cleverer Tipps für die Projektarbeit - IT-Berufe: Abschlussprüfung Teil A*. 5. Solingen : U-Form-Verlag, 2011 <http://fiae.link/ClevererTippsFuerDieProjektarbeit>. – ISBN 3882347538
- [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, Stefan Macke, versichere hiermit, dass ich meine **Dokumentation zur betrieblichen Projektarbeit** mit dem Thema

*Entwicklung von NatInfo – Webbasiertes Tool zur Unterstützung der Entwickler*

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.

Vechta, den 23.04.2015

---

STEFAN MACKE

## A Anhang

### A.1 Detaillierte Zeitplanung

<b>Analysephase</b>	<b>9 h</b>
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
<b>Entwurfsphase</b>	<b>19 h</b>
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
<b>Implementierungsphase</b>	<b>29 h</b>
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
<b>Abnahmetest der Fachabteilung</b>	<b>1 h</b>
1. Abnahmetest der Fachabteilung	1 h
<b>Einführungsphase</b>	<b>1 h</b>
1. Einführung/Benutzerschulung	1 h
<b>Erstellen der Dokumentation</b>	<b>9 h</b>
1. Erstellen der Benutzerdokumentation	2 h
2. Erstellen der Projektdokumentation	6 h
3. Programmdokumentation	1 h
3.1. Generierung durch PHPdoc	1 h
<b>Pufferzeit</b>	<b>2 h</b>
1. Puffer	2 h
<b>Gesamt</b>	<b>70 h</b>

## 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 Unified Modeling Language (UML)-Diagramme kann man auch direkt mit  $\text{\LaTeX}$  zeichnen, siehe z. B. <http://metauml.sourceforge.net/old/usecase-diagram.html>.

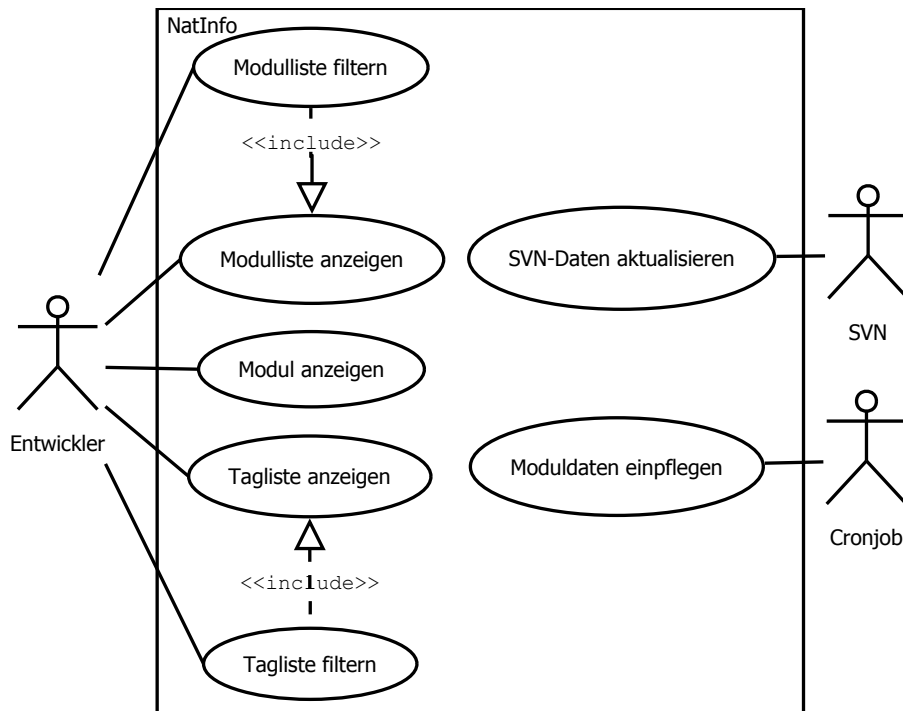


Abbildung 3: Use Case-Diagramm

### 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.
- 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 Comma Separated Value (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 Subversion (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 Natural Information System (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.

## A Anhang

### Produkteinsatz

#### 1. Anwendungsbereiche

Die Webanwendung dient als Anlaufstelle für die Entwicklung. Dort sind alle Informationen für die Module an einer Stelle gesammelt. Vorher getrennte Anwendungen werden ersetzt bzw. verlinkt.

#### 2. Zielgruppen

NatInfo wird lediglich von den Programmiersprache der Software AG (**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.5 Datenbankmodell

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

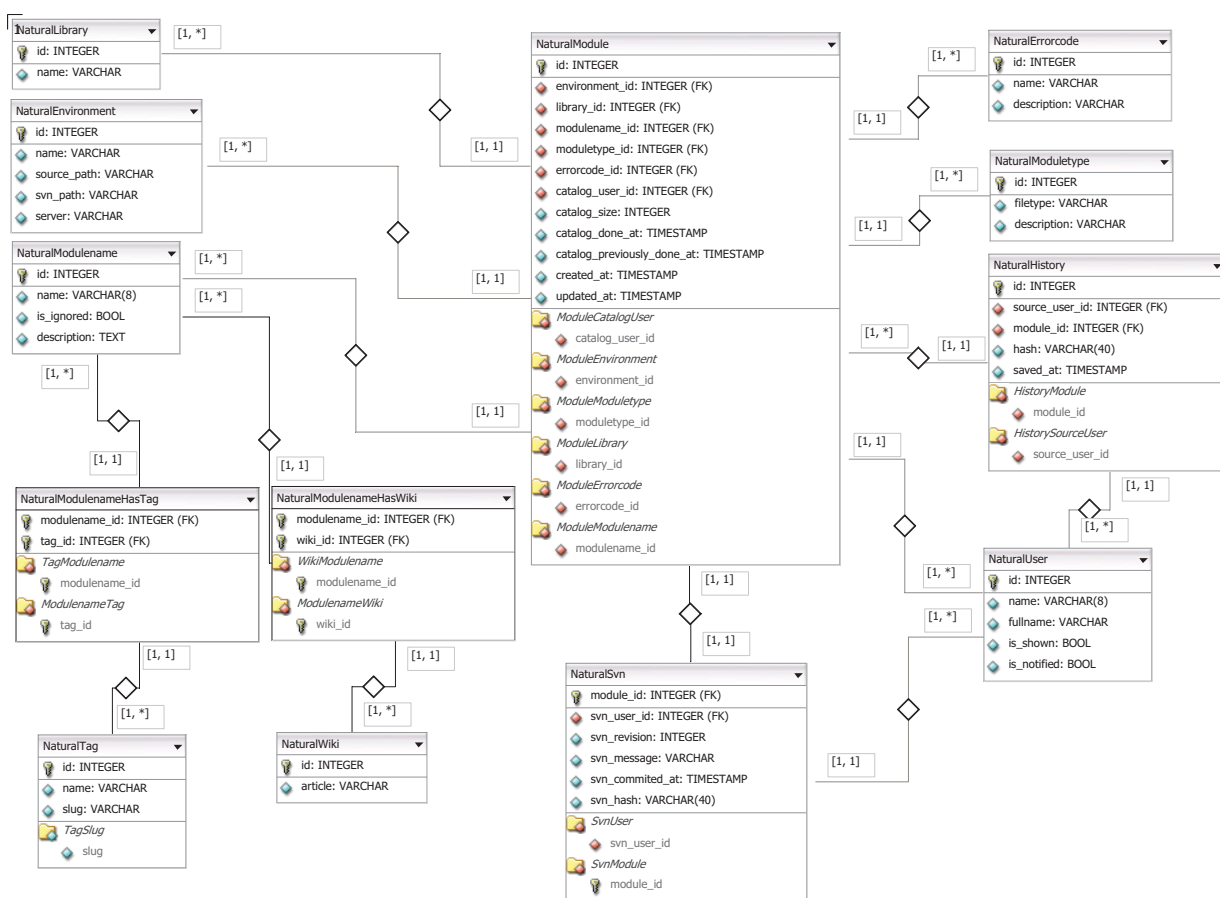


Abbildung 4: Datenbankmodell



## A.6 Oberflächenentwürfe

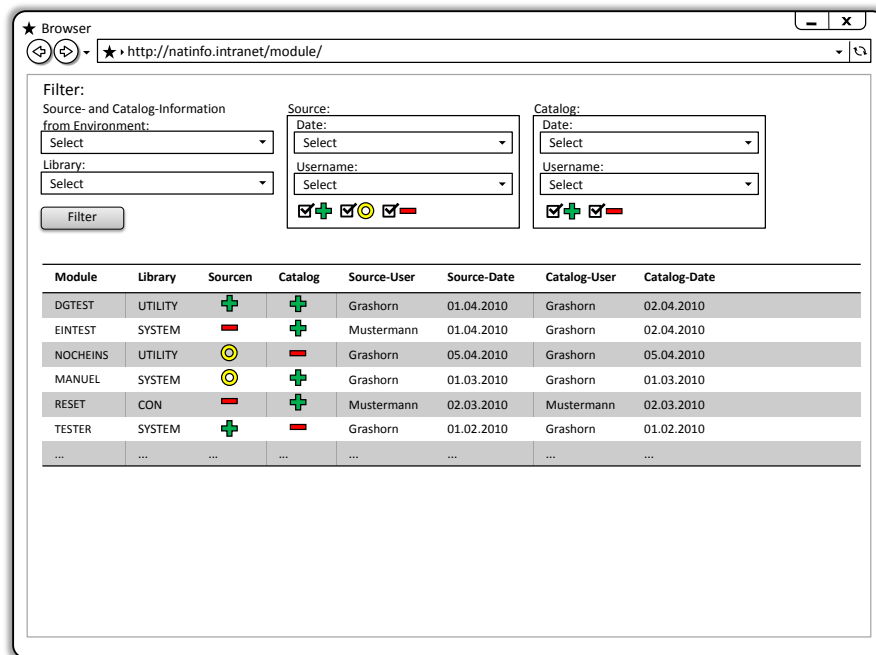


Abbildung 5: Liste der Module mit Filtermöglichkeiten

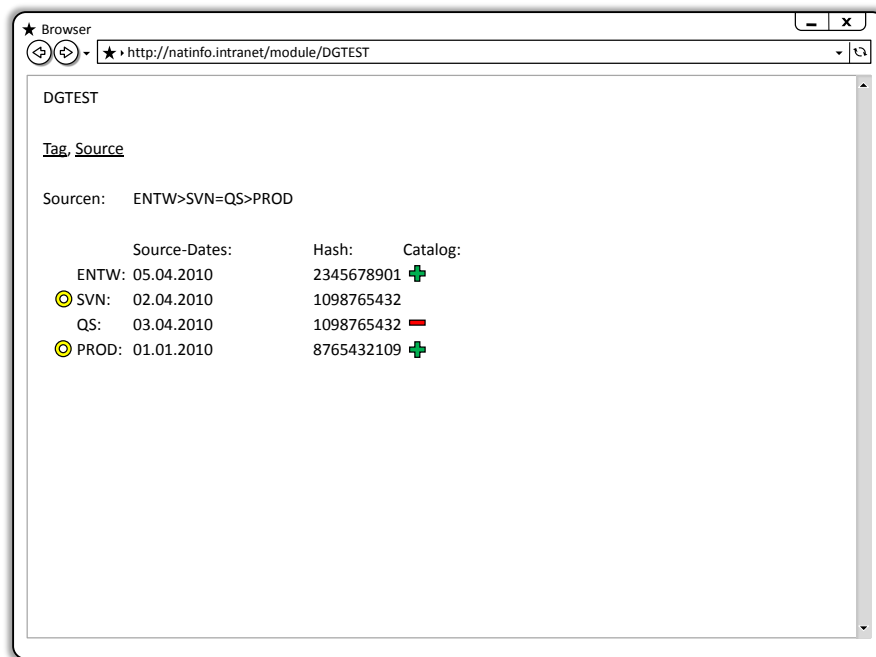


Abbildung 6: Anzeige der Übersichtsseite einzelner Module

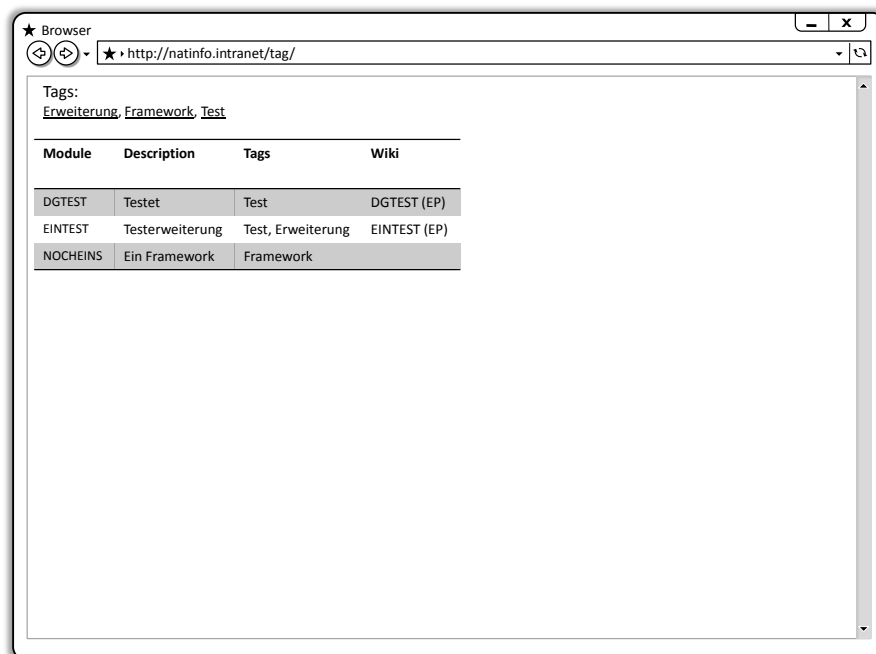


Abbildung 7: Anzeige und Filterung der Module nach Tags

## 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

<b>Environment</b>	ENTW
<b>Library</b>	Select
<b>Catalog user</b>	Select
<b>Catalog date</b>	Select
<b>Source user</b>	Select
<b>Source date</b>	Select
<a href="#">Reset</a> <a href="#">Filter</a>	

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

[\[ Top \]](#)

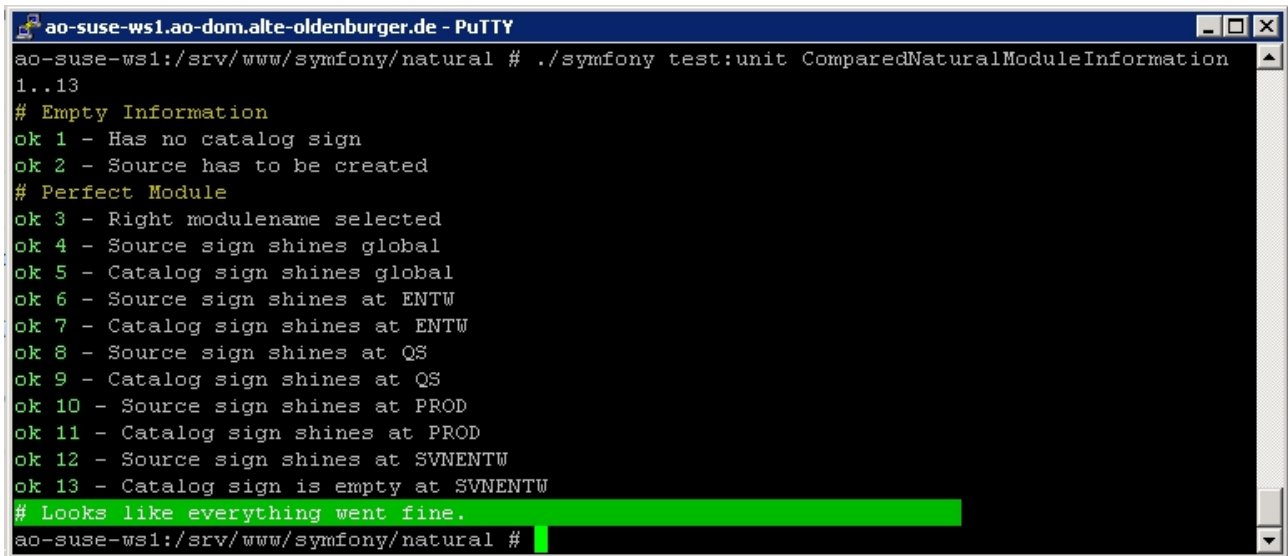
---

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

Listing 1: Testfall in PHP

```
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(NaturalmodulePeer::NAME, 'SMTAB');
14 $moduleName = NaturalmodulePeer::doSelectOne($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 ?>
```



```
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.

Listing 2: Klasse: ComparedNaturalModuleInformation

```
1 <?php
2 class ComparedNaturalModuleInformation{
3     const EMPTY_SIGN = 0;
4     const SIGN_OK = 1;
5     const SIGN_NEXT_STEP = 2;
6     const SIGN_CREATE = 3;
7     const SIGN_CREATE_AND_NEXT_STEP = 4;
8     const SIGN_ERROR = 5;
9
10    private $naturalModuleInformations = array();
11
12    public static function environments(){
13        return array("ENTW", "SVNENTW", "QS", "PROD");
14    }
15
16    public static function signOrder(){
17        return array(self::SIGN_ERROR, self::SIGN_NEXT_STEP, self::SIGN_CREATE_AND_NEXT_STEP,
18            self::SIGN_CREATE, self::SIGN_OK);
19    }
20
21    public function __construct(array $naturalInformations){
22        $this->allocateModulesToEnvironments($naturalInformations);
23        $this->allocateEmptyModulesToMissingEnvironments();
24        $this->determineSourceSignsForAllEnvironments();
25    }
26 }
```



```

25
26 private function allocateModulesToEnvironments(array $naturalInformations){
27   foreach ($naturalInformations as $naturalInformation){
28     $env = $naturalInformation->getEnvironmentName();
29     if(in_array($env, self::environments())){
30       $this->naturalModuleInformations[array_search($env, self::environments())] =
31         $naturalInformation;
32     }
33   }
34
35 private function allocateEmptyModulesToMissingEnvironments(){
36   if(array_key_exists(0, $this->naturalModuleInformations)){
37     $this->naturalModuleInformations[0]->setSourceSign(self::SIGN_OK);
38   }
39
40   for($i = 0; $i < count(self::environments()); $i++){
41     if(!array_key_exists($i, $this->naturalModuleInformations)){
42       $environments = self::environments();
43       $this->naturalModuleInformations[$i] = new EmptyNaturalModuleInformation(
44         $environments[$i]);
45       $this->naturalModuleInformations[$i]->setSourceSign(self::SIGN_CREATE);
46     }
47   }
48
49 public function determineSourceSignsForAllEnvironments(){
50   for($i = 1; $i < count(self::environments()); $i++){
51     $currentInformation = $this->naturalModuleInformations[$i];
52     $previousInformation = $this->naturalModuleInformations[$i - 1];
53     if($currentInformation->getSourceSign() <> self::SIGN_CREATE){
54       if($previousInformation->getSourceSign() <> self::SIGN_CREATE){
55         if($currentInformation->getHash() <> $previousInformation->getHash()){
56           if($currentInformation->getSourceDate('YmdHis') > $previousInformation->
57             getSourceDate('YmdHis')){
58             $currentInformation->setSourceSign(self::SIGN_ERROR);
59           }
60           else{$currentInformation->setSourceSign(self::SIGN_NEXT_STEP);}
61         }
62         else{$currentInformation->setSourceSign(self::SIGN_OK);}
63       }
64       else{$currentInformation->setSourceSign(self::SIGN_ERROR);}
65     }
66     elseif($previousInformation->getSourceSign() <> self::SIGN_CREATE &&
67       $previousInformation->getSourceSign() <> self::SIGN_CREATE_AND_NEXT_STEP){
68       $currentInformation->setSourceSign(self::SIGN_CREATE_AND_NEXT_STEP);
69     }
70   }

```

```
71 private function containsSourceSign($sign){
72     foreach($this->naturalModuleInformations as $information){
73         if($information->getSourceSign() == $sign){
74             return true;
75         }
76     }
77     return false;
78 }
79
80 private function containsCatalogSign($sign){
81     foreach($this->naturalModuleInformations as $information){
82         if($information->getCatalogSign() == $sign){
83             return true;
84         }
85     }
86     return false;
87 }
88 }
89 ?>
```

## A.11 Klassendiagramm

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

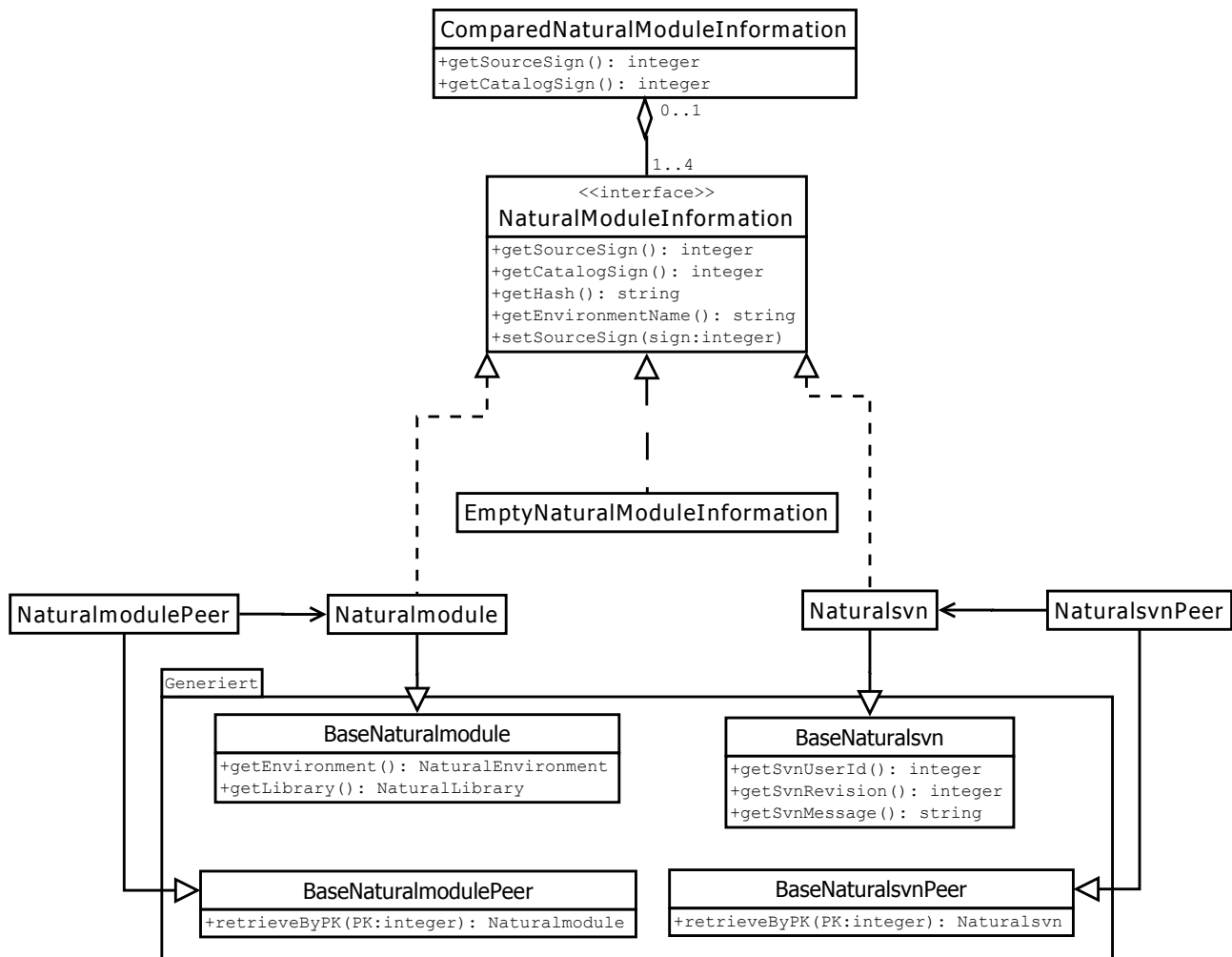







Abbildung 11: Klassendiagramm

## 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.