



Abschlussprüfung Sommer 2015

Fachinformatiker für Anwendungsentwicklung

Dokumentation zur betrieblichen Projektarbeit

## **Der Kurztitel**

### **Der Langtitel der Projektdokumentation**

Abgabedatum: Abgabeort, den 01.01.2020

#### **Prüfungsbewerber:**

Der Autor

Prüflingsstraße 1

12345 Prüflingsort

#### **Ausbildungsbetrieb:**

Das Unternehmen

Unternehmensstraße 1

12345 Unternehmensort



## DER KURZTITEL

Der Langtitel der Projektdokumentation

## Inhaltsverzeichnis

---

### Inhaltsverzeichnis

Inhaltsverzeichnis.....	I
Abbildungsverzeichnis.....	III
Tabellenverzeichnis.....	IV
Verzeichnis der Listings.....	V
Abkürzungsverzeichnis.....	VI
1 Einleitung .....	1
1.1 Projektumfeld .....	1
1.2 Projektziel .....	1
1.3 Projektbegründung .....	1
1.4 Projektschnittstellen .....	1
1.5 Projektabgrenzung .....	1
2 Projektplanung .....	1
2.1 Projektphasen .....	1
2.2 Abweichungen vom Projektantrag .....	2
2.3 Ressourcenplanung .....	2
2.4 Entwicklungsprozess.....	2
3 Analysephase.....	2
3.1 Ist-Analyse .....	2
3.2 Wirtschaftlichkeitsanalyse .....	2
3.2.1 Make or Buy-Entscheidung .....	2
3.2.2 Projektkosten .....	2
3.2.3 Amortisationsdauer .....	3
3.3 Nutzwertanalyse.....	4
3.4 Anwendungsfälle.....	4
3.5 Qualitätsanforderungen.....	4
3.6 Lastenheft/Fachkonzept .....	4
4 Entwurfsphase .....	4
4.1 Zielplattform .....	4
4.2 Architekturdesign .....	4
4.3 Entwurf der Benutzeroberfläche .....	5
4.4 Datenmodell.....	5
4.5 Geschäftslogik.....	5
4.6 Maßnahmen zur Qualitätssicherung.....	6
4.7 Pflichtenheft/Datenverarbeitungskonzept .....	6
5 Implementierungsphase .....	6
5.1 Implementierung der Datenstrukturen .....	6

## DER KURZTITEL

Der Langtitel der Projektdokumentation

## Inhaltsverzeichnis

---

5.2	Implementierung der Benutzeroberfläche .....	6
5.3	Implementierung der Geschäftslogik .....	6
6	Abnahmephase .....	7
7	Einführungsphase .....	7
8	Dokumentation .....	7
9	Fazit .....	7
9.1	Soll-/Ist-Vergleich .....	7
9.2	Lessons Learned.....	8
9.3	Ausblick.....	8
	Literaturverzeichnis .....	9
	Eidesstattliche Erklärung .....	10
	Anhang.....	i
A1	Detaillierte Zeitplanung.....	i
A2	Lastenheft (Auszug) .....	ii
A3	Use-Case-Diagramm.....	iii
A4	Pflichtenheft (Auszug) .....	iii
A5	Datenbankmodell .....	v
A6	Ereignisgesteuerte Prozesskette .....	vi
A7	Oberflächenentwürfe .....	vii
A8	Screenshots der Anwendung .....	viii
A9	Entwicklerdokumentation (Auszug) .....	x
A10	Testfall und sein Aufruf auf der Konsole .....	xi
A11	Klasse: ComparedNaturalModuleInformation .....	xii
A12	Klassendiagramm .....	xiv
A13	Benutzerdokumentation (Auszug) .....	xv

## DER KURZTITEL

Der Langtitel der Projektdokumentation

## Abbildungsverzeichnis

---

### Abbildungsverzeichnis

Abbildung 1: Use-Case-Diagramm .....	iii
Abbildung 2: Entity-Relationship-Model .....	v
Abbildung 3: Tabellenmodell .....	vi
Abbildung 4: Prozess des Einlesens eines Moduls .....	vi
Abbildung 5: Liste der Module mit Filtermöglichkeiten .....	vii
Abbildung 6: Anzeige der Übersichtsseite einzelner Module .....	viii
Abbildung 7: Anzeige und Filterung der Module nach Tags .....	viii
Abbildung 8: Liste der Module mit Filtermöglichkeiten .....	ix
Abbildung 9: Auszug aus der Entwicklerdokumentation mit <i>PHPDoc</i> .....	x
Abbildung 10: Aufruf des Testfalls auf der Konsole .....	xi
Abbildung 11: Klassendiagramm .....	xiv
Abbildung 12: Auszug aus der Benutzerdokumentation .....	xv

## DER KURZTITEL

Der Langtitel der Projektdokumentation

## Tabellenverzeichnis

---

### Tabellenverzeichnis

Tabelle 1: Grobe Zeitplanung .....	2
Tabelle 2: Kostenaufstellung .....	3
Tabelle 3: Entscheidungsmatrix.....	5
Tabelle 4: Soll-/Ist-Vergleich.....	8
Tabelle 5: Detaillierte Zeitplanung .....	ii

**DER KURZTITEL**  
Der Langtitel der Projektdokumentation

## Verzeichnis der Listings

---

### Verzeichnis der Listings

Listing 1: Testklasse.....	xii
Listing 2: Klasse <code>ComparedNaturalModuleInformation</code> .....	xiv

## DER KURZTITEL

Der Langtitel der Projektdokumentation

## Abkürzungsverzeichnis

---

### Abkürzungsverzeichnis

API .....	<i>Application Programming Interface</i>
CSV.....	<i>Comma Separated Values</i>
EPK.....	<i>Ereignisgesteuerte Prozesskette</i>
ERM .....	<i>Entity Relationship Model</i>
GUI.....	<i>Graphical User Interface</i>
HTML .....	<i>Hypertext Markup Language</i>
MVC .....	<i>Model View Controller</i>
PHP.....	<i>PHP Hypertext Preprocessor</i>
SQL.....	<i>Structured Query Language</i>
SVN.....	<i>Subversion</i>
XML.....	<i>Extensible Markup Language</i>

## **1 Einleitung**

### **1.1 Projektumfeld**

- Kurze Vorstellung des Ausbildungsbetriebs (Geschäftsfeld, Mitarbeiterzahl usw.)
- Wer ist Auftraggeber/Kunde des Projekts?

### **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**

- In welchem Zeitraum und unter welchen Rahmenbedingungen (z.B. Tagesarbeitszeit) findet das Projekt statt?
- Verfeinerung der Zeitplanung, die bereits im Projektantrag vorgestellt wurde.

---

#### **Beispiel**

---

Tabelle 1 zeigt ein Beispiel für eine grobe Zeitplanung.<sup>1</sup>

Eine detailliertere Zeitplanung ist in Tabelle 5 in Anhang A1 zu sehen.

---

<sup>1</sup> Die Beispiele in dieser Dokumentation stammen aus (Grashorn, 2010).



## DER KURZTITEL

Der Langtitel der Projektdokumentation

### Analysephase

Projektphase	Geplante Zeit
Analyse	9 h
Entwurf	20 h
Implementierung	30 h
Abnahme	1 h
Einführung	1 h
Dokumentation	9 h
<b>Gesamt</b>	<b>70 h</b>

Tabelle 1: Grobe Zeitplanung

## 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

- Rentiert sich das Projekt für das Unternehmen?

### 3.2.1 Make or Buy-Entscheidung

- Gibt es vielleicht schon ein fertiges Produkt, das 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)?

## DER KURZTITEL

Der Langtitel der Projektdokumentation

### Analysephase

#### 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 1.000 € (brutto).

$$8 \frac{\text{h}}{\text{Tag}} \cdot 220 \frac{\text{Tage}}{\text{Jahr}} = 1.760 \frac{\text{h}}{\text{Jahr}}$$

$$1.000 \frac{\text{€}}{\text{Monat}} \cdot 13,3 \frac{\text{Monate}}{\text{Jahr}} = 13.300 \frac{\text{€}}{\text{Jahr}}$$

$$\frac{13.300 \frac{\text{€}}{\text{Jahr}}}{1.760 \frac{\text{h}}{\text{Jahr}}} \approx 7,56 \frac{\text{€}}{\text{h}}$$

Es ergibt sich also ein Stundensatz von 7,56 EUR. Die Durchführungszeit des Projekts beträgt 70 Stunden. Für die Nutzung von Ressourcen<sup>2</sup> wird ein pauschaler Stundensatz von 15 EUR angenommen. Für die anderen Mitarbeiter wird pauschal ein Stundensatz von 25 EUR angenommen. Eine Aufstellung der Kosten befindet sich in Tabelle 2 und sie betragen insgesamt 2.739,20 EUR.

Vorgang	Zeit	Kosten / Stunde	Kosten
Entwicklung	70 h	7,56 € + 15 € = 22,56 €	1.579,20 €
Fachgespräch	3 h	25 € + 15 € = 40,00 €	120,00 €
Abnahme	1 h	25 € + 15 € = 40,00 €	40,00 €
Schulung	25 h	25 € + 15 € = 40,00 €	1.000,00 €
<b>Gesamt</b>			<b>2.739,20 €</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:

$$25 \cdot 220 \frac{\text{Tage}}{\text{Jahr}} \cdot 10 \frac{\text{min}}{\text{Tag}} = 55.000 \frac{\text{min}}{\text{Jahr}} \approx 917 \frac{\text{h}}{\text{Jahr}}$$

Dadurch ergibt sich eine jährliche Einsparung von:

$$917 \text{h} \cdot (25 + 15) \frac{\text{€}}{\text{h}} = 36.680 \text{ €}$$

Die Amortisationszeit beträgt also:

<sup>2</sup> Räumlichkeiten, Arbeitsplatzrechner etc.

## DER KURZTITEL

Der Langtitel der Projektdokumentation

### Entwurfsphase

---

$$\frac{2.739,20 \text{ €}}{36.680 \frac{\text{€}}{\text{Jahr}}} \approx 0,07 \text{ Jahre} \approx 4 \text{ Wochen}$$

### 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 EPK detailliert beschrieben werden.

---

#### Beispiel

---

Ein Beispiel für ein Use-Case-Diagramm findet sich im Anhang A3.

### 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 A2.

## 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. MVC).
- Ggfs. Bewertung und Auswahl von verwendeten Frameworks sowie ggfs. eine kurze Einführung in die Funktionsweise des verwendeten Frameworks.

## DER KURZTITEL

Der Langtitel der Projektdokumentation

### Entwurfsphase

#### Beispiel

Anhand der Entscheidungsmatrix in Tabelle 3 wurde für die Implementierung der Anwendung das PHP-Framework *Symfony* ausgewählt.

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

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 A7.

### 4.4 Datenmodell

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

#### Beispiel

In Anhang A5 wird ein ERM dargestellt, welches lediglich Entitäten, Relationen und die dazugehörigen Kardinalitäten enthält.

### 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?

## DER KURZTITEL

Der Langtitel der Projektdokumentation

### Implementierungsphase

---

#### Beispiel

---

Ein Klassendiagramm, welches die Klassen der Anwendung und deren Beziehungen untereinander darstellt, kann im Anhang A12 eingesehen werden.

Die EPK in Anhang A6 zeigt den grundsätzlichen Ablauf beim Einlesen eines Moduls.

#### 4.6 Maßnahmen zur Qualitätssicherung

- Welche Maßnahmen werden ergriffen, um die Qualität des Projektergebnisses (siehe Kapitel 3.5) 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) aufbauende Pflichtenheft ist im Anhang A4 zu finden.

## 5 Implementierungsphase

### 5.1 Implementierung der Datenstrukturen

- Beschreibung der angelegten Datenbank (z.B. Generierung von SQL aus Modellierungswerkzeug oder händisches Anlegen), XML-Schemas usw.

### 5.2 Implementierung der Benutzeroberfläche

- Beschreibung der Implementierung der Benutzeroberfläche, falls dies separat zur Implementierung der Geschäftslogik erfolgt (z.B. bei 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 A8.

### 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: Es 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.

## DER KURZTITEL

Der Langtitel der Projektdokumentation

### Abnahmephase

#### Beispiel

Die Klasse `ComparedNaturalModuleInformation` findet sich im Anhang A11.

## 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 A10. Dort ist auch der Aufruf des Tests auf der Konsole des Webserver 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, 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 A13.

Die Entwicklerdokumentation wurde mittels *PHPDoc* automatisch generiert. Ein beispielhafter Auszug aus der Dokumentation einer Klasse findet sich im Anhang A9.

## 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 4 zu erkennen ist, konnte die Zeitplanung bis auf wenige Ausnahmen eingehalten werden.

## DER KURZTITEL

Der Langtitel der Projektdokumentation

### Fazit

---

Phase	Geplant	Tatsächlich	Differenz
Analyse	9 h	10 h	+1 h
Entwurf	20 h	20 h	
Implementierung	30 h	27 h	-3 h
Abnahme	1 h	1 h	
Einführung	1 h	1 h	
Dokumentation	9 h	11 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)?

## DER KURZTITEL

Der Langtitel der Projektdokumentation

## Literaturverzeichnis

---

### Literaturverzeichnis

Grashorn, D., 2010. *Entwicklung von NatInfo – Webbasiertes Tool zur Unterstützung der Entwickler*, Vechta: s.n.

ISO/IEC 9126-1, 2001. *Software-Engineering – Qualität von Software-Produkten – Teil 1: Qualitätsmodell*. s.l.:s.n.



## DER KURZTITEL

Der Langtitel der Projektdokumentation

## Eidesstattliche Erklärung

---

### Eidesstattliche Erklärung

Ich, Der Autor, versichere hiermit, dass ich meine Dokumentation zur betrieblichen Projektarbeit mit dem Thema

*Der Kurztitel – Der Langtitel der Projektdokumentation*

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.

Abgabeort, den 03.10.2016

---

DER AUTOR

## Anhang

### A1 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>20 h</b>
1. Prozessentwurf	3 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>30 h</b>
1. Anlegen der Datenbank	1 h
2. Umsetzung der HTML-Oberflächen und Stylesheets	5 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 geparsten Metadaten	3 h
3.8. Erstellen einer Übersichtsseite für ein einzelnes Modul	1 h
4. Nächtlichen Batchjob einrichten	1 h

## DER KURZTITEL

Der Langtitel der Projektdokumentation

### Anhang

<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 <i>PHPdoc</i>	1 h
<b>Gesamt</b>	<b>70 h</b>

Tabelle 5: Detaillierte Zeitplanung

## A2 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.

### A3 Use-Case-Diagramm

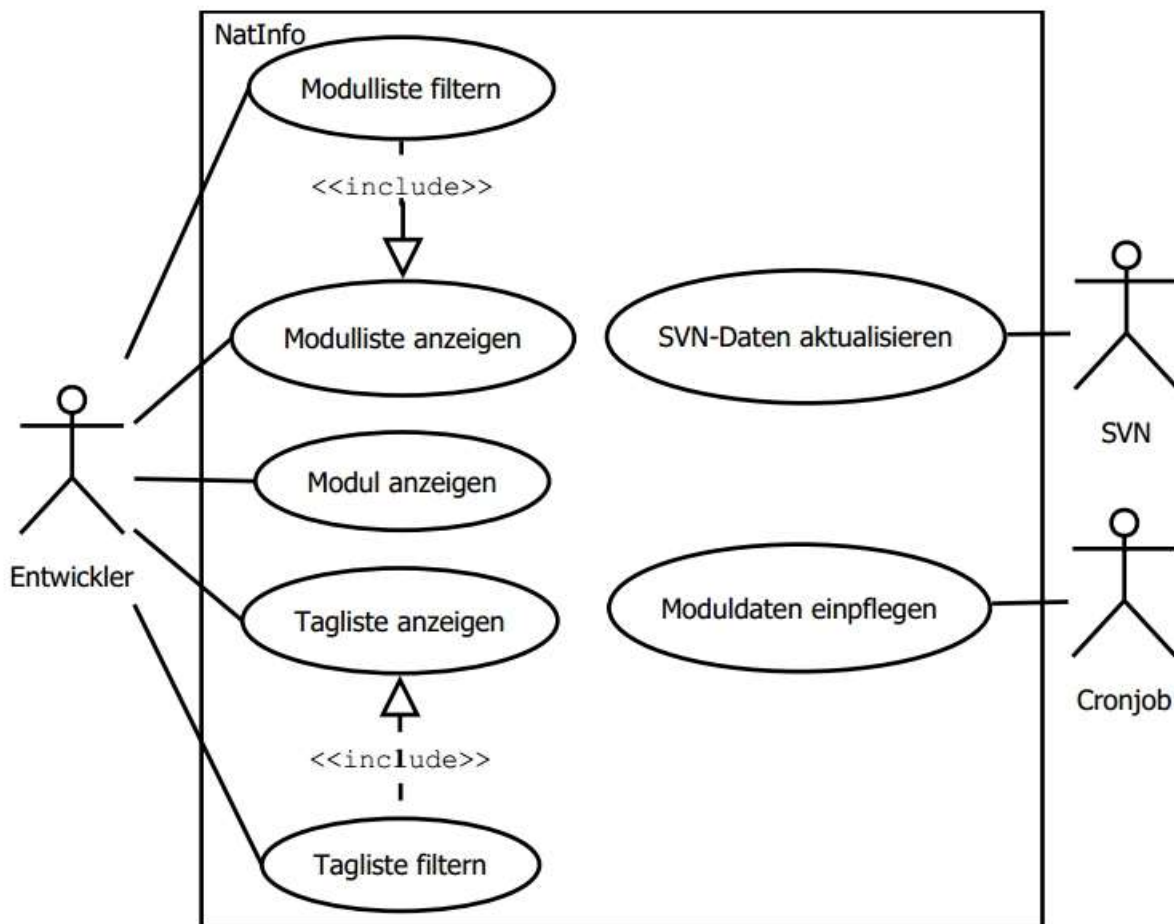


Abbildung 1: Use-Case-Diagramm

### A4 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 tätigen Schritt.

## DER KURZTITEL

Der Langtitel der Projektdokumentation

## Anhang

---

- 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 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 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

1.1. 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

2.1. NatInfo wird lediglich von den Natural-Entwicklern in der EDV-Abteilung genutzt.

## DER KURZTITEL

Der Langtitel der Projektdokumentation

## Anhang

### 3. Betriebsbedingungen

3.1. 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.

### A5 Datenbankmodell

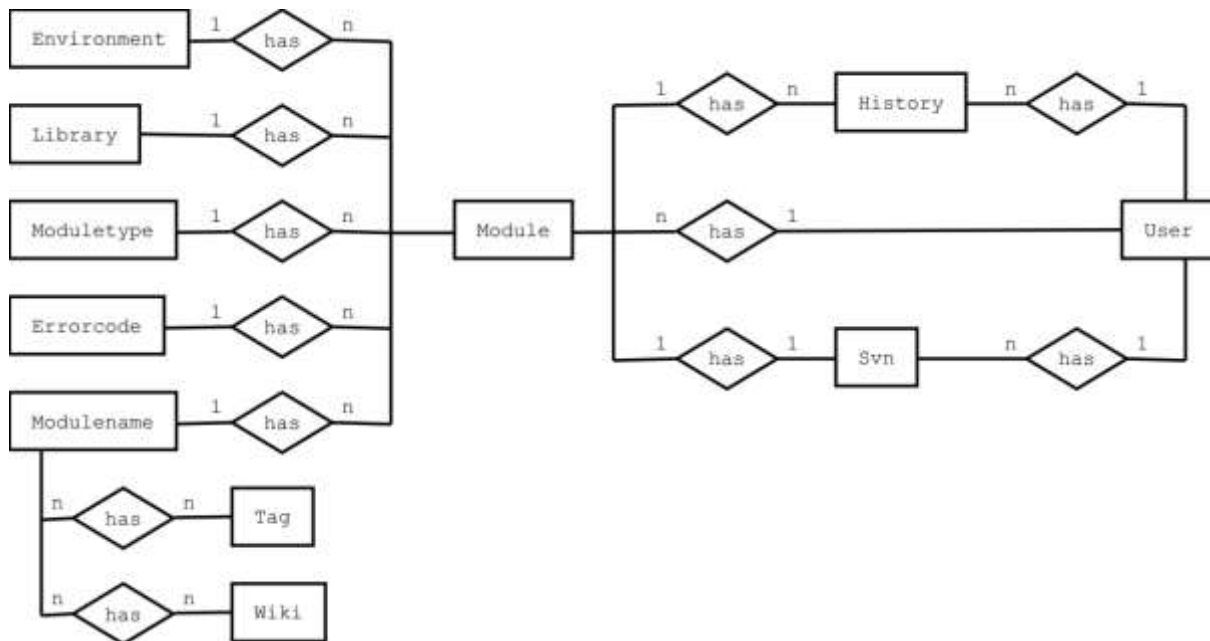
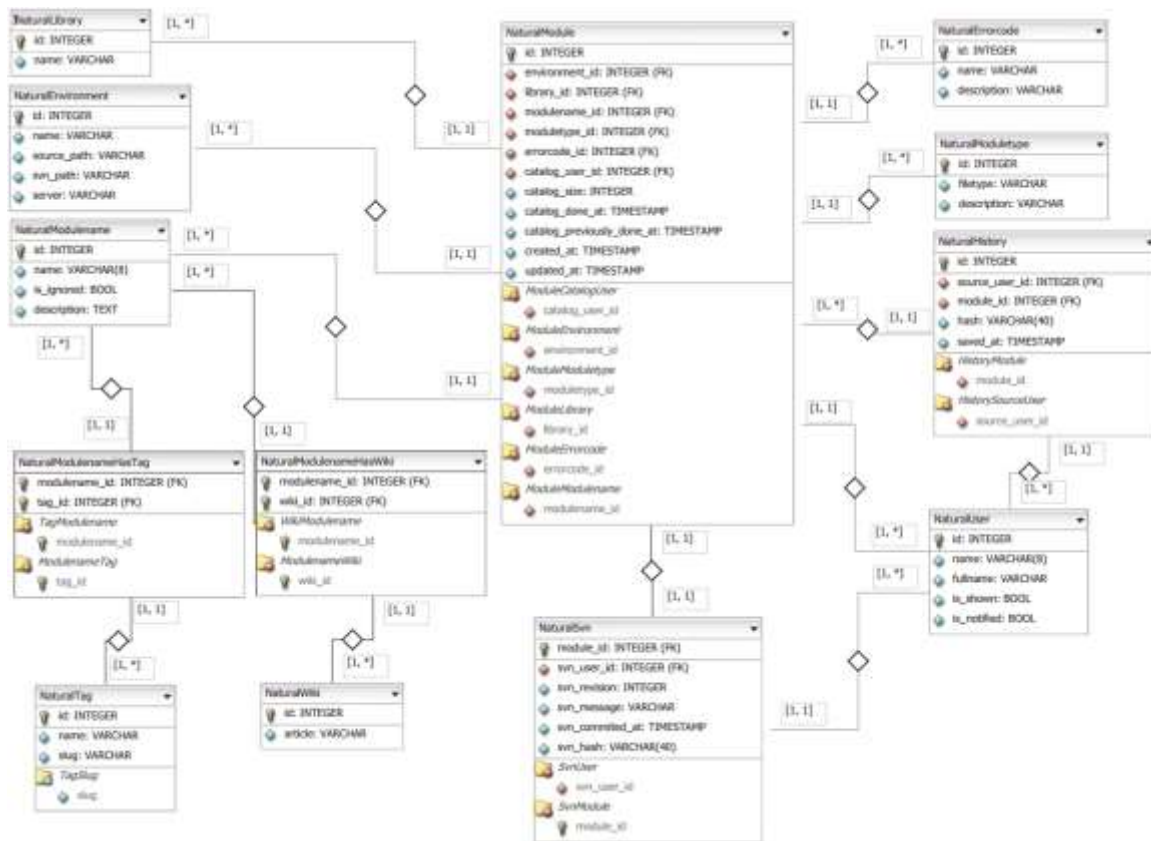


Abbildung 2: Entity-Relationship-Model

## Anhang



### Abbildung 3: Tabellenmodell

## A6 Ereignisgesteuerte Prozesskette



#### Abbildung 4: Prozess des Einlesens eines Moduls

## A7 Oberflächenentwürfe

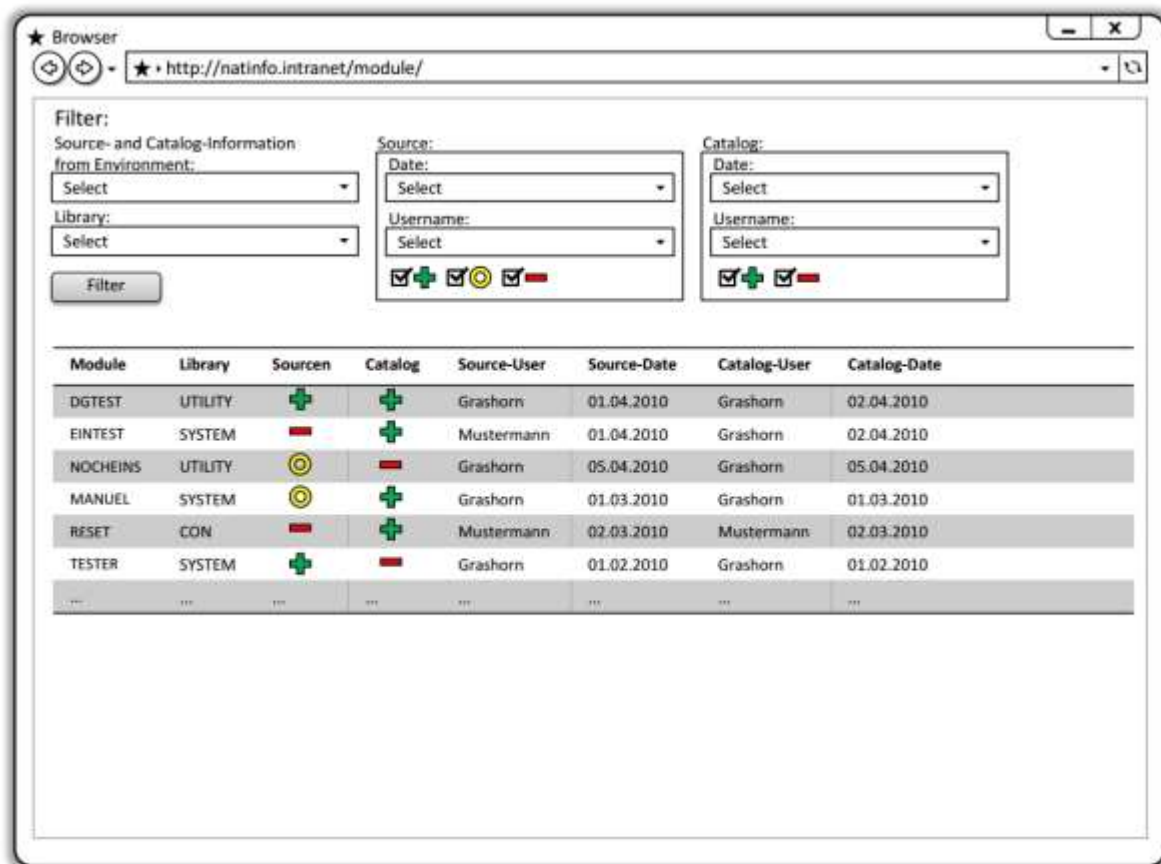


Abbildung 5: Liste der Module mit Filtermöglichkeiten



## DER KURZTITEL

Der Langtitel der Projektdokumentation

## Anhang

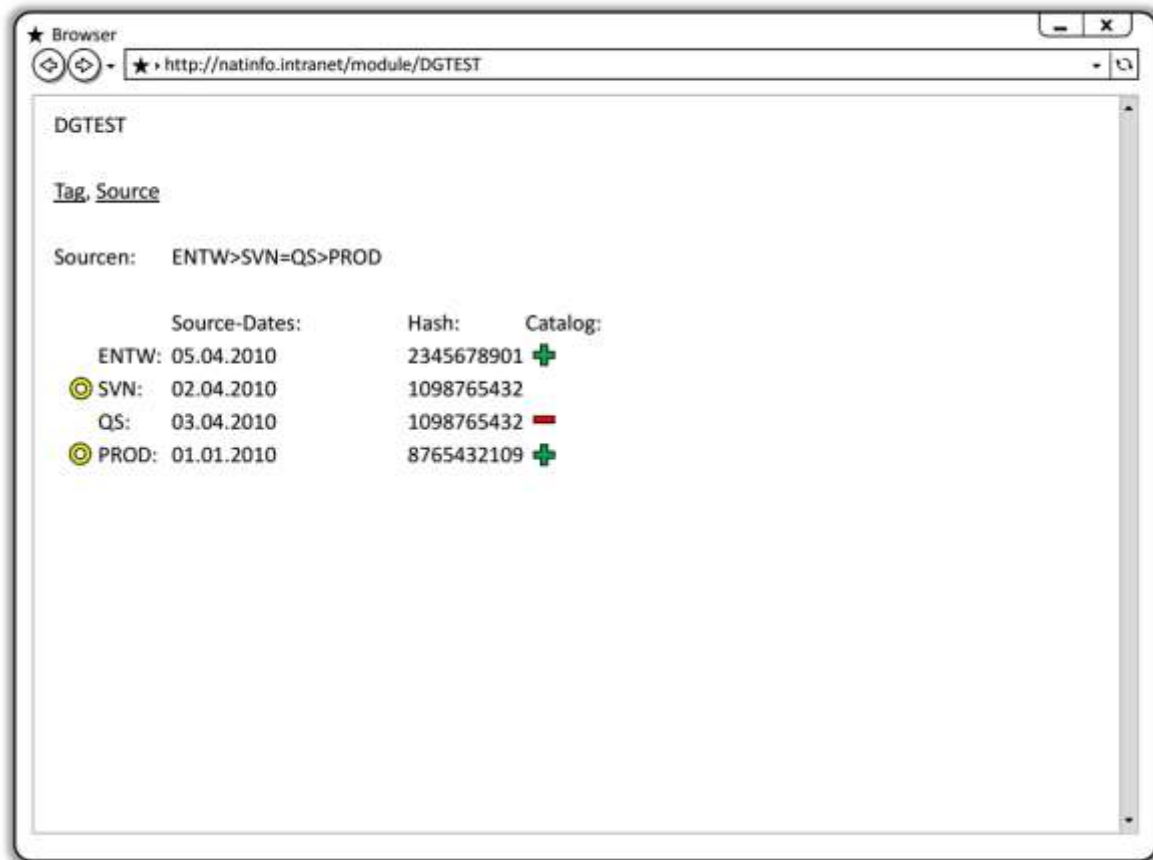


Abbildung 6: Anzeige der Übersichtsseite einzelner Module

## A8 Screenshots der Anwendung



Abbildung 7: Anzeige und Filterung der Module nach Tags

# DER KURZTITEL

## Der Langtitel der Projektdokumentation

### Anhang



The screenshot shows the NATINFO web application. The header features the NATINFO logo and a navigation bar with a home icon, "/ Modules / ENTW", and "Tags, Modules". On the right, there are links for "Häufig benötigte Seiten", "Ein-Klick-Menü", "Direktaufruf", and "Inhaltsuche". The main content area is titled "Modules" and contains a filter panel on the left with dropdown menus for Environment (ENTW), Library (Select), Catalog user (Select), Catalog date (Select), Source user (Select), and Source date (Select). Below these are "Reset" and "Filter" buttons. To the right is a table with 8 columns: Name, Library, Source, Catalog, Source-User, Source-Date, Catalog-User, and Catalog-Date. The table lists five modules: SMTAB, DGTAB, DGTEST, OHNETAG, and OHNEWIKI, each with associated library, source, catalog, user, and date information.

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 8: Liste der Module mit Filtermöglichkeiten

## A9 Entwicklerdokumentation (Auszug)

**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.

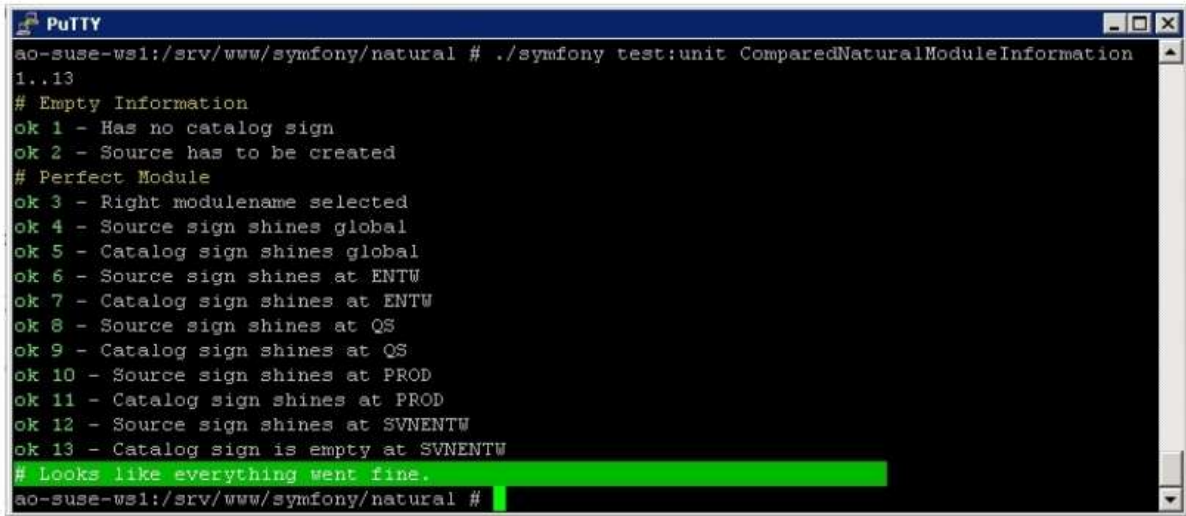
**Tags:**

**see:** parent::\_\_construct()  
**access:** public

[\[ Top \]](#)

Abbildung 9: Auszug aus der Entwicklerdokumentation mit *PHPDoc*

## A10 Testfall und sein Aufruf auf der Konsole



```

ao-suse-wsl:/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-wsl:/srv/www/symfony/natural #

```

Abbildung 10: Aufruf des Testfalls auf der Konsole

```

$t->comment('Empty Information');
$emptyComparedInformation = new
    ComparedNaturalModuleInformation(array());
$t->is($emptyComparedInformation->getCatalogSign(),
    ComparedNaturalModuleInformation::EMPTY_SIGN, 'Has no catalog
    sign');
$t->is($emptyComparedInformation->getSourceSign(),
    ComparedNaturalModuleInformation::SIGN_CREATE, 'Source has to be
    created');

$t->comment('Perfect Module');
$criteria = new Criteria();
$criteria->add(NaturalmodulePeer::NAME, 'SMTAB');
$moduleName = NaturalmodulePeer::doSelectOne($criteria);
$t->is($moduleName->getName(), 'SMTAB', 'Right modulename
    selected');
$comparedInformation = $moduleName->loadNaturalModuleInformation();
$t->is($comparedInformation->getSourceSign(),
    ComparedNaturalModuleInformation::SIGN_OK, 'Source sign shines
    global');
$t->is($comparedInformation->getCatalogSign(),
    ComparedNaturalModuleInformation::SIGN_OK, 'Catalog sign shines
    global');
$infos = $comparedInformation->getNaturalModuleInformations();
foreach($infos as $info) {

```

```

$env = $info->getEnvironmentName();

$t->is($info->getSourceSign(),
ComparedNaturalModuleInformation::SIGN_OK, 'Source sign shines
at ' . $env);

if($env != 'SVNENTW') {

    $t->is($info->getCatalogSign(),
ComparedNaturalModuleInformation::SIGN_OK, 'Catalog sign shines
at ' . $info->getEnvironmentName());

} else {

    $t->is($info->getCatalogSign(),
ComparedNaturalModuleInformation::EMPTY_SIGN, 'Catalog sign is
empty at ' . $info->getEnvironmentName());

}

}

```

**Listing 1: Testklasse**

## **A11 Klasse: ComparedNaturalModuleInformation**

Kommentare und simple Getter/Setter werden nicht gezeigt.

```

class ComparedNaturalModuleInformation {

    const EMPTY_SIGN = 0;

    ...

    const SIGN_ERROR = 5;

    private $naturalModuleInformations = array();

    public static function environments() {

        return array("ENTW", "SVNENTW", "QS", "PROD");

    }

    public static function signOrder() {

        return array(self::SIGN_ERROR, self::SIGN_NEXT_STEP,
self::SIGN_CREATE_AND_NEXT_STEP, self::SIGN_CREATE,
self::SIGN_OK);

    }

    public function __construct(array $naturalInformations) {

        $this->allocateModulesToEnvironments($naturalInformations);

        $this->allocateEmptyModulesToMissingEnvironments();

        $this->determineSourceSignsForAllEnvironments();

    }

}

```

```

    }

    private function allocateModulesToEnvironments(array
    $naturalInformations) {
        foreach ($naturalInformations as $naturalInformation) {
            $env = $naturalInformation->getEnvironmentName();
            if(in_array($env, self::environments())) {
                $this->naturalModuleInformations[array_search($env,
                self::environments())] = $naturalInformation;
            }
        }
    }

    private function allocateEmptyModulesToMissingEnvironments() {
        if(array_key_exists(0, $this->naturalModuleInformations)) {
            $this->naturalModuleInformations[0]-
            >setSourceSign(self::SIGN_OK);
        }

        for($i = 0;$i < count(self::environments());$i++) {
            if(!array_key_exists($i, $this-
            >naturalModuleInformations)) {
                $environments = self::environments();
                $this->naturalModuleInformations[$i] = new
                EmptyNaturalModuleInformation($environments[$i]);
                $this->naturalModuleInformations[$i]-
                >setSourceSign(self::SIGN_CREATE);
            }
        }
    }

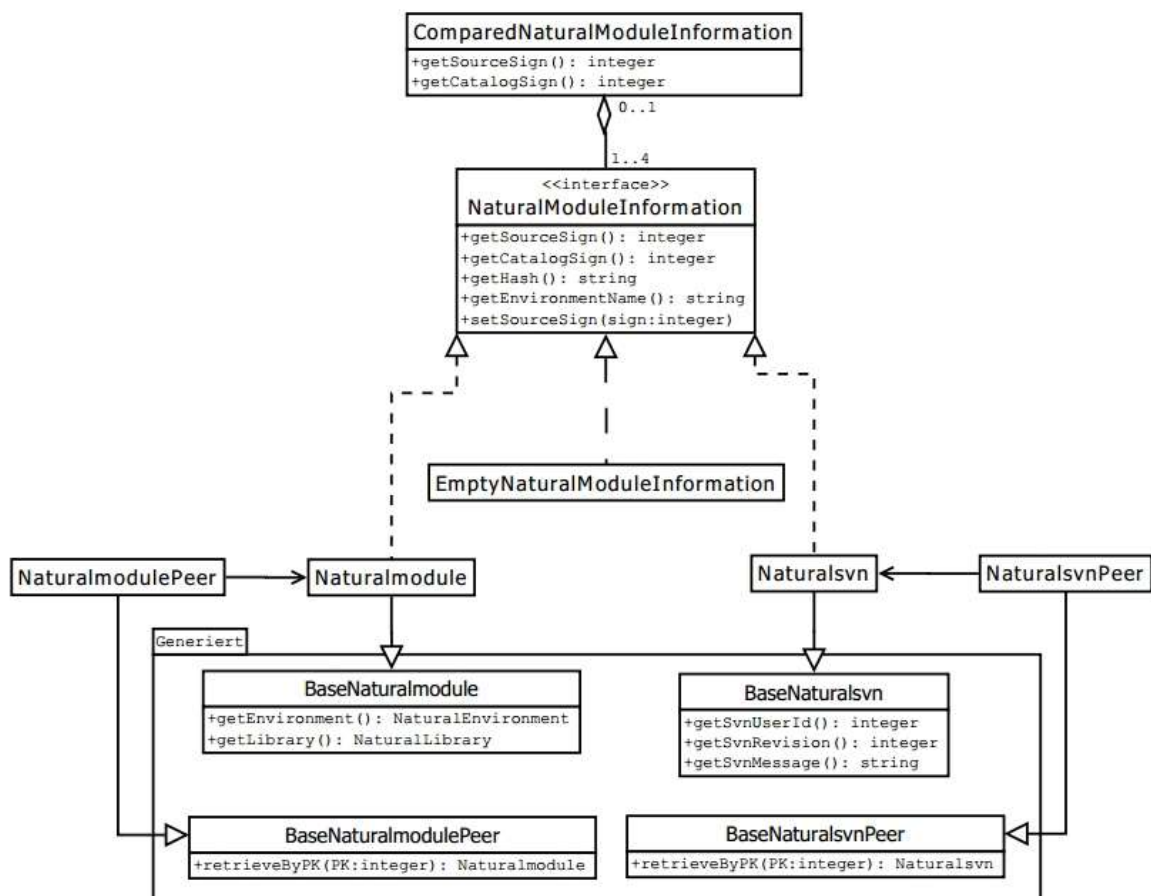
    private function containsSourceSign($sign) {
        foreach($this->naturalModuleInformations as $information) {
            if($information->getSourceSign() == $sign) {
                return true;
            }
        }
        return false;
    }

```

```
private function containsCatalogSign($sign) {
    foreach($this->naturalModuleInformations as $information) {
        if($information->getCatalogSign() == $sign) {
            return true;
        }
    }
    return false;
}
```

### Listing 2: Klasse `ComparedNaturalModuleInformation`






## A12 Klassendiagramm



### Abbildung 11: Klassendiagramm



## A13 Benutzerdokumentation (Auszug)

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.

**Abbildung 12: Auszug aus der Benutzerdokumentation**