

Abschlussprüfung Sommer 2023 Fachinformatiker für Anwendungsentwicklung

Dokumentation zur betrieblichen Projektarbeit

**Entwurf und Entwicklung für ein Compilerplugin das Codingkonventionen und andere Vereinbarungen Während der Kompilierung prüft.**

Abgabedatum: Braunschweig, den 16.04.2023

**Prüfungsbewerber:**

Hauke Wolf

Hinter den Gärten 34

38729 Langelsheim

**Ausbildungsbetrieb:**

CGI Deutschland B.V. &Co. KG

Am Alten Bahnhof 13

38122 Braunschweig



# Inhaltsverzeichnis

Inhalt

[Inhaltsverzeichnis I](#_Toc129267867)

[Tabellenverzeichnis III](#_Toc129267868)

[Verzeichnis der Listings IV](#_Toc129267869)

[Abkürzungsverzeichnis V](#_Toc129267870)

[1 Einleitung 1](#_Toc129267871)

[1.1 Projektumfeld 1](#_Toc129267872)

[1.2 Projektziel 1](#_Toc129267873)

[1.3 Projektbegründung 1](#_Toc129267874)

[1.4 Projektschnittstellen 1](#_Toc129267875)

[1.5 Projektabgrenzung 1](#_Toc129267876)

[2 Projektplanung 1](#_Toc129267877)

[2.1 Projektphasen 1](#_Toc129267878)

[2.2 Abweichungen vom Projektantrag 2](#_Toc129267879)

[2.3 Ressourcenplanung 2](#_Toc129267880)

[3 Analysephase 2](#_Toc129267881)

[3.1 Ist-Analyse 2](#_Toc129267882)

[3.2 Wirtschaftlichkeitsanalyse 2](#_Toc129267883)

[3.2.1 Make or Buy-Entscheidung 2](#_Toc129267884)

[3.2.2 Projektkosten 2](#_Toc129267885)

[3.2.3 Amortisationsdauer 3](#_Toc129267886)

[3.3 Nutzwertanalyse 4](#_Toc129267887)

[3.4 Anwendungsfälle 4](#_Toc129267888)

[3.5 Qualitätsanforderungen 4](#_Toc129267889)

[3.6 Lastenheft/Fachkonzept 4](#_Toc129267890)

[4 Entwurfsphase 4](#_Toc129267891)

[4.1 Zielplattform 4](#_Toc129267892)

[4.2 Architekturdesign 4](#_Toc129267893)

[4.3 Entwurf der Benutzeroberfläche 5](#_Toc129267894)

[4.4 Datenmodell 5](#_Toc129267895)

[4.5 Geschäftslogik 5](#_Toc129267896)

[4.6 Maßnahmen zur Qualitätssicherung 6](#_Toc129267897)

[4.7 Pflichtenheft/Datenverarbeitungskonzept 6](#_Toc129267898)

[5 Implementierungsphase 6](#_Toc129267899)

[5.1 Implementierung der Datenstrukturen 6](#_Toc129267900)

[5.2 Implementierung der Benutzeroberfläche 6](#_Toc129267901)

[5.3 Implementierung der Geschäftslogik 6](#_Toc129267902)

[6 Abnahmephase 7](#_Toc129267903)

[7 Einführungsphase 7](#_Toc129267904)

[8 Dokumentation 7](#_Toc129267905)

[9 Fazit 8](#_Toc129267906)

[9.1 Soll-/Ist-Vergleich 8](#_Toc129267907)

**Abbildungsverzeichnis**

[Abbildung 1: Use-Case-Diagramm iii](#_Toc129263760)

[Abbildung 2: Entity-Relationship-Modell vi](#_Toc129263761)

[Abbildung 3: Tabellenmodell vi](#_Toc129263762)

[Abbildung 4: Prozess des Einlesens eines Moduls vi](#_Toc129263763)

[Abbildung 5: Liste der Module mit Filtermöglichkeiten viii](#_Toc129263764)

[Abbildung 6: Anzeige der Übersichtsseite einzelner Module viii](#_Toc129263765)

[Abbildung 7: Anzeige und Filterung der Module nach Tags ix](#_Toc129263766)

[Abbildung 8: Liste der Module mit Filtermöglichkeiten ix](#_Toc129263767)

[Abbildung 9: Auszug aus der Entwicklerdokumentation mit *PHPDoc* x](#_Toc129263768)

[Abbildung 10: Aufruf des Testfalls auf der Konsole xi](#_Toc129263769)

[Abbildung 11: Klassendiagramm xvi](#_Toc129263770)

[Abbildung 12: Auszug aus der Benutzerdokumentation xvii](#_Toc129263771)

# Tabellenverzeichnis

**Tabellenverzeichnis**

[Tabelle 1: Grobe Zeitplanung 1](#_Toc129267809)

[Tabelle 2: Kostenaufstellung 3](#_Toc129267810)

[Tabelle 3: Entscheidungsmatrix 5](#_Toc129267811)

[Tabelle 4: Soll-/Ist-Vergleich 9](#_Toc129267812)

[Tabelle 5: Detaillierte Zeitplanung ii](#_Toc129267813)

# 

# Verzeichnis der Listings

**Verzeichnis der Listings**

Listing 1: Testklasse ............................................................................................................. xii

Listing 2: Klasse ComparedNaturalModuleInformation ............................................. xiv

# Abkürzungsverzeichnis

**Abkürzungsverzeichnis**

API ........... *Application Programming Interface*

CSV .......... *Comma Separated Values*

EPK .......... *Ereignisgesteuerte Prozesskette*

ERM ......... *Entity Relationship Model*

GUI........... *Graphical User Interface*

HTML ....... *Hypertext Markup Language*

MVC ......... *Model View Controller*

PHP .......... *PHP Hypertext Preprocessor*

SQL .......... *Structured Query Language*

SVN .......... *Subversion*

XML .......... *Extensible Markup Langua*

|  |
| --- |
| 1 Einleitung |

## Projektumfeld

Das Projekt wird im Rahmen der Ausbildung zum Fachinformatiker Fachrichtung

Anwendungsentwicklung durch den Autor für die CGI Deutschland B.V. & Co. KG, genauer den Standort Braunschweig, umgesetzt. Die CGI Deutschland B.V. & Co. KG ist die deutsche Niederlassung der CGI Inc., eines kanadischen Unternehmens und globaler Dienstleister für IT und Geschäftsprozesse. Die CGI Inc. verfügt mit 400 Standorten in 40 Ländern über insgesamt 90000 Mitarbeiter, davon 5000+ Mitarbeiter in Deutschland aus 30 verschiedenen Städten. Die Ausbildung fand in der Abteilung der Softwareentwicklung am Standort Braunschweig statt.

## 1.2 Projektziel

Es soll ein Plugin für den Kotlin Compiler entwickelt werden, welches die Einhaltung von erstellten Regeln sicherstellt. Es soll möglich sein das Tool für verschiedene Vorgaben und Regeln anzuwenden. Ein theoretisches Beispiel für eine Regel könnte die Verpflichtung sein, dass zu jedem Fragment auch immer eine dazugehörige View existieren muss. Sind die Vorgaben bzw. Regeln nicht erfüllt wird das Programm, während dem Kompilieren dies mit Hilfe des Plugins erkennen und dementsprechend nicht kompilieren oder eine Warnung ausgeben.

Dies wiederum bietet den Vorteil, dass nicht das gesamte Projekt gebaut werden muss und somit Zeit in der Entwicklung gespart wird. Außerdem wird so mithilfe des Plugins langfristig die Projektintegrität verbessert.

## 1.3 Projektbegründung

Ein Compiler Tool dieser Art ist in vielen Fällen eine Bereicherung für die CGI. Es werden mithilfe des Tools nicht nur die Entwicklungszeiten an sich verbessert, sondern auch die langfristige Projektintegrität. Die verbesserte Projektintegrität führt dazu, dass innerhalb des Projekts kürzere und bessere Entwicklungszyklen entstehen was Zeit und auch Geld einsparen kann. Durch eine konstante Umsetzung der von dem Tool gegebenen Konventionen werden langfristig auch Fehler in der Projektstrucktur minimiert.

## 1.4 Projektschnittstellen

Das Projekt wird als Libery Tool bereitgestellt. So kann es Problemlos ohne großen Aufwand in die IDE Android Studio eigebunden und benutzt werden. Es ist also nötig das Android Studios von den anwendenden Teams für die Entwicklung verwendet wird.

Ein Entwicklerteam der CGI kann völlig unabhängig das Tool einbinden und es nach gewünschtem umfang benutzen.

Die Abnahme des Projektes übernimmt aufgrund der Ausbildungssituation der Praktische Ausbilder Paul Nicolaus Schuberth.

## 1.5 Projektabgrenzung

Das Projekt wird in dem Sinne begrenzt das der Umfang der zu prüfenden Konventionen begrenzt wird. Auch ist die Produktivsetzung nicht Teil des Projektumfangs. Es soll dennoch eine grundlegende Funktionalität innerhalb eines Testprogrammes vorhanden sein.

|  |
| --- |
| 2 Projektplanung |

## 2.1 Projektphasen

Der Umfang des Projekts beträgt 80 Stunden. Im Rahmen der Planung des Projektantrags wurde bereits eine grobe Verteilung der Stunden auf die einzelnen Projektphasen vorgenommen. Die Projektbearbeitung fand zwischen dem 25.02.2023 bis zum 19.04.2023 statt. Das Projekt wurde in der Regel Montags, Dienstags und Donnerstag und Freitags bearbeitet.¹ Die Arbeitszeit pro Tag betrug Ca. 7,5h.

Eine grobe Planung ist nachfolgend bereitgestellt. Die Hauptphasen wurden in weitere Unterpunkte aufgeteilt. Die Planung mit Detaillierten Unterpunkten ist in Anhang [*A1 Detaillierte Zeitplanung*](#Anhang_1_Detaillierte_Zeitplanung) hinterlegt.

|  |  |
| --- | --- |
| **Projektphase** | **Geplante Zeit** |
| Analyse | 9 h |
| Entwurf | 12 h |
| Implementierung | 32 h |
| Abnahme | 4 h |
| Einführung | 4 h |
| Dokumentation | 19 h |

**Gesamt 80 h**

Tabelle 1: Grobe Zeitplanung

¹ An einem Mittwoch ist in der Regel die Berufsschule besucht wurden

## 

## 2.2 **Abweichungen vom Projektantrag**

*Noch keine*

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

## 2.3 **Ressourcenplanung**

Die für dieses Projekt verwendeten Ressourcen sind im Anhang A.3: Verwendete Ressourcen zu finden. Die Hardware wurde bereits von CGI bereitgestellt und bei der Software wurden bereits Lizenzen verwendet. Als Räumlichkeiten wurde das Büro in Braunschweig genutzt und gelegentlich auch das Homeoffice des Autors. Eine Ausführlichere Auflistung der verwendeten Ressourcen sind in [A2 Verwendete Ressourcen](#Anhang_2_Verwendete_Ressourcen) aufgelistet.

**2.4** Entwicklungsprozess

Bei der Entwicklung des Projektes wird auf das Wasserfall Prinzip zurückgegriffen. Der Autor hat sich dafür entschieden da es in dem Umfang des Projektes nicht nötig ist Agil vorzugehen, da dies die Komplexität für einen Entwickler unnötig vergrößern würde. Das Wasserfall Prinzip hingegen erlaubt es Simple und schnelle Schritte in der Entwicklung der Software zu gehen.

Das Projekt wird am Ende mit Unit Tests abgedeckt um eine sichere und zuverlässige Funktionalität zu gewährleisten.

|  |
| --- |
| 3 Analysephase |

## 3.1 **Ist-Analyse**

Aktuell wird bei der CGI noch kein einheitliches Tooling für das Durchsetzen von Programmiere Konventionen eingesetzt.

Es soll ein Kotlin Compiler Plugin entwickelt werden, welches die Einhaltung von erstellten Regeln sicherstellt, so können aktuelle Projekte langfristig verbessert werden.

In einigen Projekt der CGI werden keine einheitlichen Paradigmen verfolgt, stattdessen wurden neue Komponenten nach aktuellem Stand der Technik umgesetzt, ältere Teile aber nicht konsequent aktualisiert.

Es ist also dringend nötig ein solches Tool einzuführen um den Entwicklungsprozess deutlich zu verbessern.

## 

## 3.2 **Wirtschaftlichkeitsanalyse**

Wie bereits in Kapitelabschnitt [1.3 Projektbegründung](#_1.3__Projektbegründung) und [3.1 Ist-Zustandsanalyse](#_3.1__Ist-Analyse) begründet, ist das Tool unbedingt vonnöten um den aktuellen Standards von Kunden und Technik gerecht zu werden. Jedoch stellt sich die Frage, ob diese Entwicklung auch wirtschaftlich Sinnvoll ist. Dies wird in den folgenden Abschnitten überprüft und begründet.

### 

### 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)?

**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).

h Tage h 8 · 220 = 1.760 Tag Jahr Jahr

€ Monate €

1.000 · 13, 3 = 13.300 Monat Jahr Jahr

€

13.300 𝐽𝑎ℎ𝑟 €

≈ 7,56

h

1.760 hJahr

Es ergibt sich also ein Stundensatz von *7,56 EUR*. Die Durchführungszeit des Projekts beträgt 70 Stunden. Für die Nutzung von Ressourcen[[1]](#footnote-1) 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 €* |

***Gesamt 2.739,20 €***

Tabelle 2: Kostenaufstellung

### 3.2.3 Amortisationsdauer

* Welche monetären Vorteile bietet das Projekt (z.B. Einsparung von Lizenzkosten, Arbeitszeitersparnis, 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:



Jahr Tag Jahr Jahr

Dadurch ergibt sich eine jährliche Einsparung von:

€

917h · (25 + 15) = 36.680 € h Die Amortisationszeit beträgt also:

**Entwurfsphase**

2.739,20 €

≈ 0, 07 𝐽𝑎ℎ𝑟𝑒 ≈ 4 𝑊𝑜𝑐ℎ𝑒𝑛

€

36.680

𝐽𝑎ℎ𝑟

## 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.

**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, XMLSchemas) 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?

**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.

**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 Webservers 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**

|  |
| --- |
| 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.

**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 |

**Gesamt 70 h 70 h**

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**

**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.

**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**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Analysephase** |  |  | | **9 h** |
| 1. Analyse des Ist-Zustands |  | 3 h | |  |
| 1.1.1 Fachgespräch mit der EDV-Abteilung | 1 h |  | |  |
| 1.1.2 Prozessanalyse | 2 h |  | |  |
| 1.2 „Make or buy“-Entscheidung und Wirtschaftlichkeitsanalyse |  | 1 h | |  |
| 1.3 Erstellen eines Use-Case-Diagramms |  | 2 h | |  |
| 1.4 Erstellen des Lastenhefts mit der EDV-Abteilung |  | 3 h | |  |
| **Entwurfsphase** |  |  | | **12 h** |
| 2.1 Prozessentwurf |  | 4 h | |  |
| 2.2 Benutzeroberflächen entwerfen und abstimmen |  | 1 h | |  |
| 2.3 Erstellen eines UML-Komponentendiagramms der Anwendung |  | 3 h | |  |
| 2.4 Erstellen des Pflichtenhefts |  | 4 h | |  |
| **Implementierungsphase** |  |  | | **32 h** |
| 3. Programmierung der KSP-Module für die Regeln |  | 32 h | |  |
| 3.1. Aufsetzen eines Testprogrammes | 2 h |  | |  |
| 3.2. Einrichten von Abhängigkeiten und Funktionen | 3 h |  | |  |
| 3.3. Implementierung der Grundstruktur für KSP | 4 h |  | |  |
| 3.4. Erstellung von KSP Verarbeitung | 12 h |  | |  |
| 3.5. Regeln für KSP anpassen und Verbessern | 5 h |  | |  |
| 3.6. Erstellen von KSP Regeln für unterschiedliche Aspekte | 6 h |  | |  |
| **Abnahmetest der Fachabteilung** |  | |  | **4 h** |
| 4. Testen der Anwendung |  | | 3 h |  |
| 4.1.1 Fehler überarbeiten und aufarbeiten |  | | 3 h |  |
| 4.1 Abnahme durch Ausbilder |  | | 1 h |  |
| **Einführungsphase** |  | |  | **4 h** |
| 1. Einführung/Benutzerschulung |  | | 4 h |  |
| **Erstellen der Dokumentation** |  | |  | **19 h** |
| 1. Erstellen der Benutzerdokumentation/Readme |  | | 3 h |  |
| 2. Erstellen der Projektdokumentation |  | | 15 h | |
| 3. Programmdokumentation |  | | 1 h | |
|  |  | |  | |

**Gesamt 80 h** Tabelle 5: Detaillierte Zeitplanung

**A2 Verwendete Ressourcen**

|  |  |  |  |
| --- | --- | --- | --- |
| **Ressourcentyp** | **Ressourcenname** | **Version ?** | **Verwendungszweck** |
| System | Windows |  | Betriebssystem des Verwendeten Rechners |
| Anwendungen | Android Studios |  | Entwicklungsumgebung |
| Buildmanagement | Gradle |  | Buildmanagement für das Projekt |
|  | Git/Github |  | Versionirungsverwaltung |
| Personal | Entwickler/Auszubildender |  | Entwicklung des Projektes |
|  | Ausbilder |  | Codereview |
| Hardware | Laptop |  | Laptop für die Enwicklungsumgebung |

Tabelle 6: Verwendete Ressourcen

**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. Die Anwendung muss die von Subversion und einem externen Programm bereitgestellten Informationen (z.B. Source-Benutzer, -Datum, Hash) verarbeiten.
   2. Auslesen der Beschreibung und der Stichwörter aus dem Sourcecode.
2. Darstellung der Daten
   1. Die Anwendung muss eine Liste aller Module erzeugen inkl. Source-Benutzer und Datum, letztem Commit-Benutzer und -Datum für alle drei Umgebungen.
   2. Verknüpfen der Module mit externen Tools wie z.B. Wiki-Einträgen zu den Modulen oder dem Sourcecode in Subversion.
   3. Die Sourcen der Umgebungen müssen verglichen und eine schnelle Übersicht zur Einhaltung des allgemeinen Entwicklungsprozesses gegeben werden.
   4. Dieser Vergleich muss auf die von einem bestimmten Benutzer bearbeiteten Module eingeschränkt werden können.
   5. Die Anwendung muss in dieser Liste auch Module anzeigen, die nach einer Bearbeitung durch den gesuchten Benutzer durch jemand anderen bearbeitet wurden.
   6. Abweichungen sollen kenntlich gemacht werden.
   7. Anzeigen einer Übersichtsseite für ein Modul mit allen relevanten Informationen zu diesem.
3. Sonstige Anforderungen
   1. Die Anwendung muss ohne das Installieren einer zusätzlichen Software über einen Webbrowser im Intranet erreichbar sein.
   2. Die Daten der Anwendung müssen jede Nacht bzw. nach jedem SVN-Commit automatisch aktualisiert werden.
   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).
   4. Die Anwendung soll jederzeit erreichbar sein.
   5. Da sich die Entwickler auf die Anwendung verlassen, muss diese korrekte Daten liefern und darf keinen Interpretationsspielraum lassen.
   6. Die Anwendung muss so flexibel sein, dass sie bei Änderungen im Entwicklungsprozess einfach angepasst werden kann.

**A3 Use-Case-Diagramm**

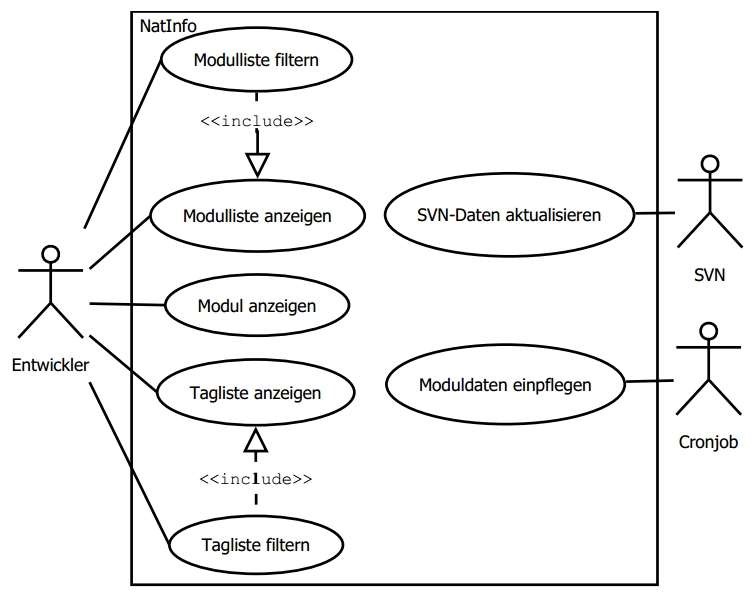


Abbildung 1: Use-Case-Diagramm

**A4 Pflichtenheft (Auszug)**

**Zielbestimmung**

1. Musskriterien
   1. 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.
   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.
   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.
   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.
   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.
   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. 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
   1. NatInfo wird lediglich von den Natural-Entwicklern in der EDV-Abteilung genutzt.
3. Betriebsbedingungen
   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**

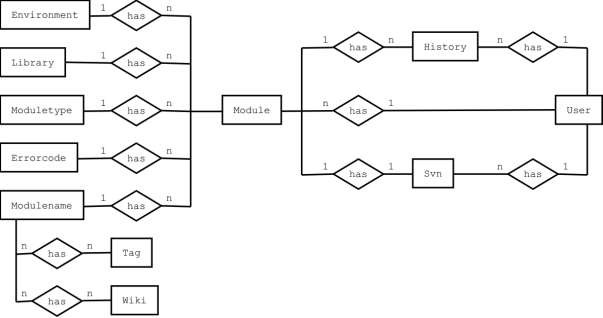


Abbildung 2: Entity-Relationship-Modell

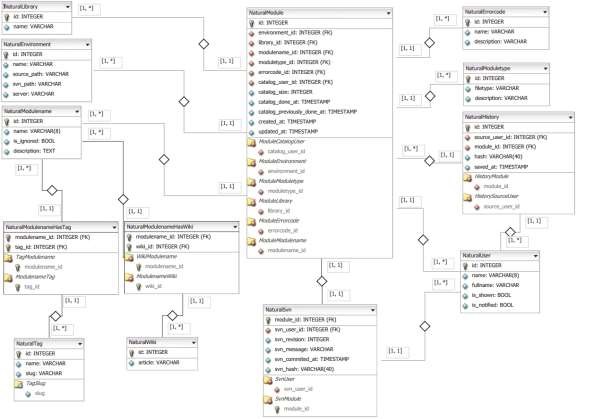


Abbildung 3: Tabellenmodell

**A6 Ereignisgesteuerte Prozesskette**

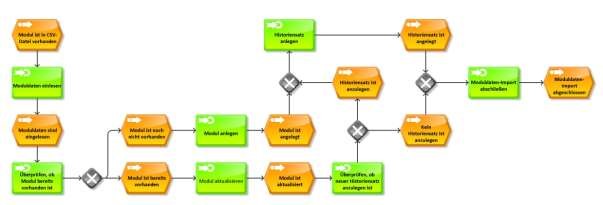


Abbildung 4: Prozess des Einlesens eines Moduls

**A7 Oberflächenentwürfe**

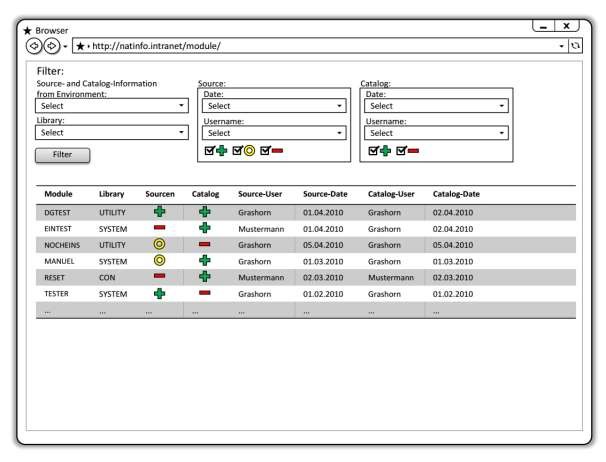


Abbildung 5: Liste der Module mit Filtermöglichkeiten

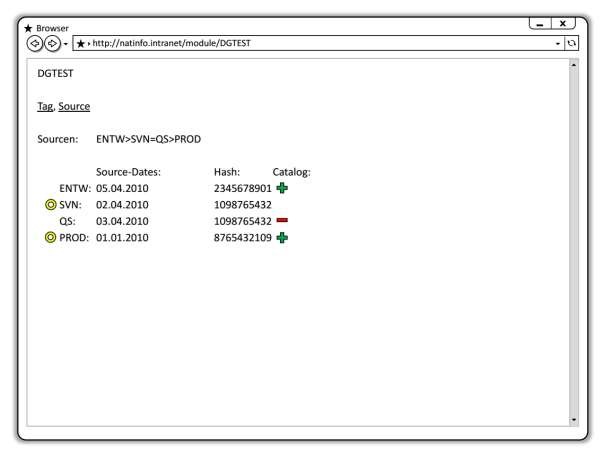


Abbildung 6: Anzeige der Übersichtsseite einzelner Module

**A8 Screenshots der Anwendung**



Abbildung 7: Anzeige und Filterung der Module nach Tags



Abbildung 8: Liste der Module mit Filtermöglichkeiten

**A9 Entwicklerdokumentation (Auszug)**

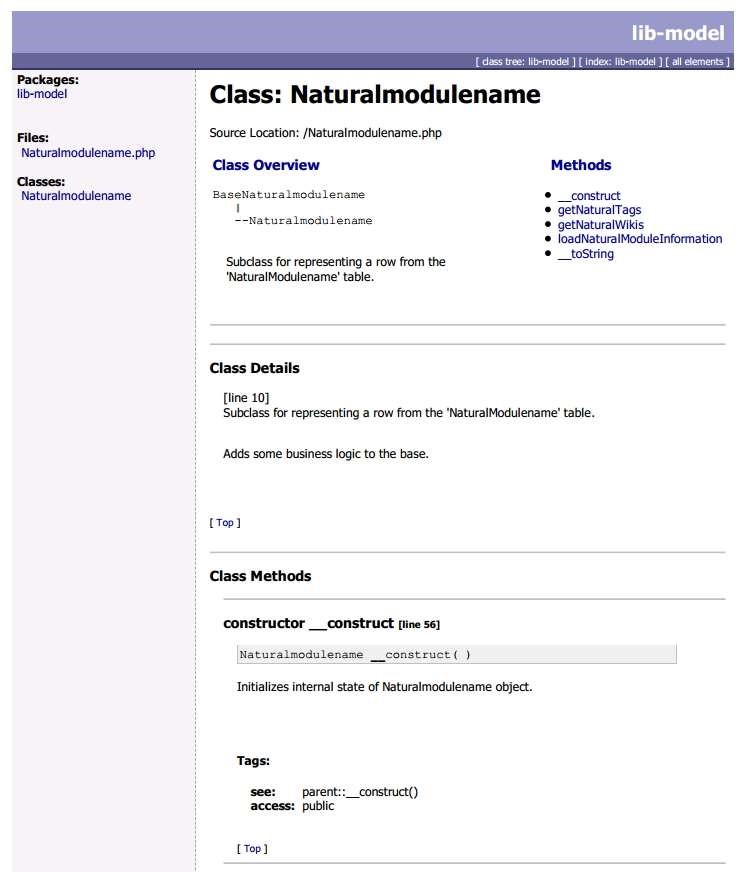


Abbildung 9: Auszug aus der Entwicklerdokumentation mit *PHPDoc*

**A10 Testfall und sein Aufruf auf der Konsole**

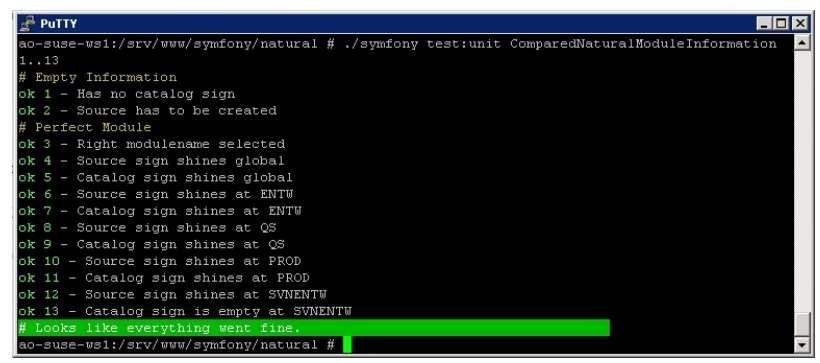


Abbildung 10: Aufruf des Testfalls auf der Konsole

|  |  |
| --- | --- |
| $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(NaturalmodulenamePeer::NAME, 'SMTAB');  $moduleName = NaturalmodulenamePeer::doSelectOne($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() == $sign) {  return true;  }  }  return false; | | | |
|  | } priv | ate function containsCatalogSign($sign) {  foreach($this->naturalModuleInformations as $information) { if($information->getCatalogSign() == $sign) { return true;  } | |
|  |  | } retu | rn false; |
| } | } | | |

**Listing 2: Klasse ComparedNaturalModuleInformation**

**A12 Klassendiagramm**

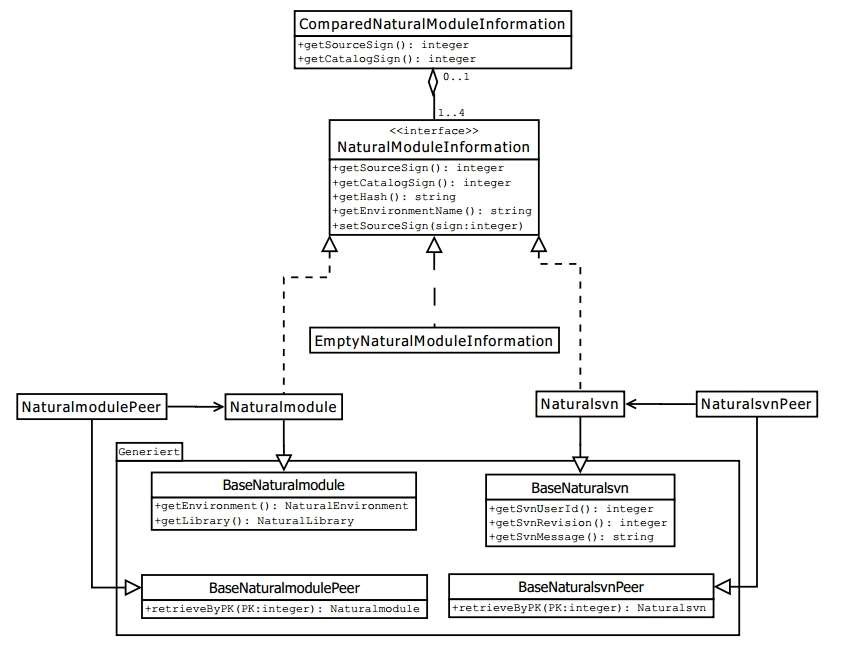


Abbildung 11: Klassendiagramm

**A13 Benutzerdokumentation (Auszug)**

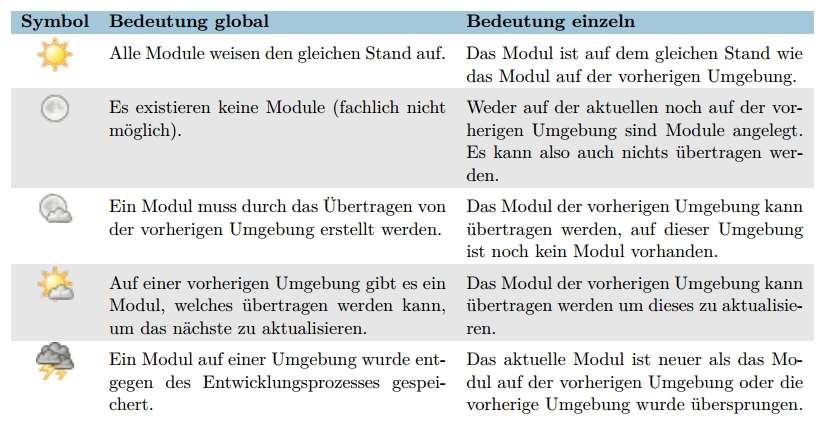


Abbildung 12: Auszug aus der Benutzerdokumentation

1. Räumlichkeiten, Arbeitsplatzrechner etc. [↑](#footnote-ref-1)