Abschlussprüfung Sommer 2023

Fachinformatiker für Anwendungsentwicklung

Dokumentation zur betrieblichen Projektarbeit

**Eigenes Logo**

Entwicklung eines 2D – „Schiffe versenken“ Spiels  
zur Simulation einer Seeschlacht

Abgabedatum:

Neckargemünd,

den 25.08.2023

**Prüfungsbewerber:**

Denis Ojdanic

Martinstraße 4

89518 Heidenheim

**Ausbildungsbetrieb:**

SRH Berufbildungswerk Neckargemünd GmbH

Im Spitzerfeld 25

69151 Neckargemünd

Inhaltsverzeichnis

## Inhaltsverzeichnis

[Inhaltsverzeichnis 2](#_Toc139623900)

[**Abbildungsverzeichnis** 3](#_Toc139623901)

[**Tabellenverzeichnis** 4](#_Toc139623902)

[**Verzeichnis der Listings** 4](#_Toc139623903)

[**Abkürzungsverzeichnis** 4](#_Toc139623904)

[1 Einleitung 5](#_Toc139623905)

[1.1 Projektumfeld 5](#_Toc139623906)

[1.2 Projektziel 6](#_Toc139623907)

[1.3 Projektbegründung 7](#_Toc139623908)

[1.4 Projektschnittstellen 7](#_Toc139623909)

[1.5 Projektabgrenzung 7](#_Toc139623910)

[2 Projektplanung 7](#_Toc139623911)

[2.1 Projektphasen 8](#_Toc139623912)

[2.2 Abweichungen vom Projektantrag 8](#_Toc139623913)

[2.3 Ressourcenplanung 8](#_Toc139623914)

[2.4 Entwicklungsprozess 9](#_Toc139623915)

[3 Analysephase 9](#_Toc139623916)

[3.1 Ist-Analyse 9](#_Toc139623917)

[3.2 Wirtschaftlichkeitsanalyse 9](#_Toc139623918)

[3.1.1 Make or Buy-Entscheidung 9](#_Toc139623919)

[3.1.2 Projektkosten 9](#_Toc139623920)

[3.1.3 Amortisationsdauer 10](#_Toc139623921)

[3.3 Nutzwertanalyse 11](#_Toc139623922)

[3.4 Anwendungsfälle 11](#_Toc139623923)

[3.5 Qualitätsanforderungen 11](#_Toc139623924)

[3.6 Lastenheft/Fachkonzept 12](#_Toc139623925)

[4 Entwurfsphase 12](#_Toc139623926)

[4.1 Zielplattform 12](#_Toc139623927)

[4.2 Architekturdesign 12](#_Toc139623928)

[4.3 Entwurf der Benutzeroberfläche 13](#_Toc139623929)

[4.4 Datenmodell 13](#_Toc139623930)

[4.5 Geschäftslogik 13](#_Toc139623931)

[4.6 Maßnahmen zur Qualitätssicherung 14](#_Toc139623932)

[4.7 Pflichtenheft/Datenverarbeitungskonzept 14](#_Toc139623933)

[5 Implementierungsphase 14](#_Toc139623934)

[5.1 Implementierung der Datenstrukturen 14](#_Toc139623935)

[5.2 Implementierung der Benutzeroberfläche 15](#_Toc139623936)

[5.3 Implementierung der Geschäftslogik 15](#_Toc139623937)

[6 Abnahmephase 16](#_Toc139623938)

[7 Einführungsphase 16](#_Toc139623939)

[8 Dokumentation 16](#_Toc139623940)

[9 Fazit 17](#_Toc139623941)

[9.1 Soll-/Ist-Vergleich 17](#_Toc139623942)

[9.2 Lessons Learned 18](#_Toc139623943)

[9.3 Ausblick 18](#_Toc139623944)

[10 Literaturverzeichnis 18](#_Toc139623945)

[11 Anhang 18](#_Toc139623946)

[A1 Detaillierte Zeitplanung 18](#_Toc139623947)

[A2 Lastenheft (Auszug) 20](#_Toc139623948)

[A3 Use-Case-Diagramm 21](#_Toc139623949)

[A4 Pflichtenheft (Auszug) 21](#_Toc139623950)

[A5 Datenbankmodell 23](#_Toc139623951)

[A6 Ereignisgesteuerte Prozesskette 24](#_Toc139623952)

[A7 Oberflächenentwürfe 25](#_Toc139623953)

[A8 Screenshots der Anwendung 26](#_Toc139623954)

[A9 Entwicklerdokumentation (Auszug) 27](#_Toc139623955)

[A10 Testfall und sein Aufruf auf der Konsole 28](#_Toc139623956)

[A11 Klasse: ComparedNaturalModuleInformation 30](#_Toc139623957)

[A12 Klassendiagramm 33](#_Toc139623958)

[A13 Benutzerdokumentation (Auszug) 33](#_Toc139623959)

**Abbildungsverzeichnis**

## Abbildungsverzeichnis

[Abbildung 1: Use-Case-Diagramm iii](#_bookmark56)

[Abbildung 2: Entity-Relationship-Model v](#_bookmark59)

[Abbildung 3: Tabellenmodell vi](#_bookmark60)

[Abbildung 4: Prozess des Einlesens eines Moduls vi](#_bookmark62)

[Abbildung 5: Liste der Module mit Filtermöglichkeiten vii](#_bookmark64)

[Abbildung 6: Anzeige der Übersichtsseite einzelner Module viii](#_bookmark65)

[Abbildung 7: Anzeige und Filterung der Module nach Tags viii](#_bookmark67)

[Abbildung 8: Liste der Module mit Filtermöglichkeiten ix](#_bookmark68)

[Abbildung 9: Auszug aus der Entwicklerdokumentation mit *PHPDoc* x](#_bookmark70)

[Abbildung 10: Aufruf des Testfalls auf der Konsole xi](#_bookmark72)

[Abbildung 11: Klassendiagramm xiv](#_bookmark77)

[Abbildung 12: Auszug aus der Benutzerdokumentation xv](#_bookmark79)

**Tabellenverzeichnis**

## Tabellenverzeichnis

[Tabelle 1: Grobe Zeitplanung 2](#_bookmark13)

[Tabelle 2: Kostenaufstellung 3](#_bookmark23)

[Tabelle 3: Entscheidungsmatrix 5](#_bookmark31)

[Tabelle 4: Soll-/Ist-Vergleich 8](#_bookmark46)

[Tabelle 5: Detaillierte Zeitplanung ii](#_bookmark53)

## Verzeichnis der Listings

## Verzeichnis der Listings

[Listing 1: Testklasse xii](#_bookmark73)

[Listing 2: Klasse ComparedNaturalModuleInformation xiv](#_bookmark75)

**Abkürzungsverzeichnis**

## Abkürzungsverzeichnis

API Application Programming Interface

CSV Comma Separated Values

EPK Ereignisgesteuerte Prozesskette

GUI Graphical User Interface

HTML Hypertext Markup Language

CSS Cascading Style Sheets

JS JavaScript

PHP PHP Hypertext Preprocessor

JSON JavaScript Object Notation

PAP Programmablaufplan

XML Extensible Markup Language

**Einleitung**

1. **Einleitung**

## Projektumfeld

Text

Der Auftraggeber ist eine Förderschule mit Förderschwerpunkt körperliche und motorische Entwicklung in Heidenheim. Die Schule bietet Kindern die Möglichkeit an, einen Hauptschulabschluss zu erwerben. Diese Schule hat ca. 200 Schüler insgesamt, unterrichtet grundsätzlich aber in kleinen Klassengrößen, um besser auf die individuellen Bedürfnisse der Schüler eingehen zu können. Das Ziel ist es, die Schüler in ihrer Entwicklung in vollen Umfang zu unterstützen und sie somit auf das spätere Leben trotz motorischen Einschränkungen vorzubereiten.

Der Kontakt ist durch einen Freund meines Vaters entstanden, der Lehrer dieser Schule ist. Er hat mich bei der Schulleitung empfohlen. Diese hatte das Vorhaben bezüglich des Projekts „Schiffe versenken“ danach bei mir in Auftrag gegeben.

Das Projekt soll in Zukunft im EDV-Unterricht zum Einsatz kommen, um mit motorisch beeinträchtigten Personen an die Handkoordination anhand der Mausbewegung zu sensibilisieren. Die Demoversion wird auf einen lokalliegenden Webserver erstellt und getestet, bevor es dort in Zukunft regulär zum Einsatz kommt.

Das Projekt wird im Rahmen der betrieblichen Projektarbeit in der SRH Neckargemünd erstellt.

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

## Projektziel

Text

Die Förderschule Ojde-Schule hat mich beauftragt, ein „Schiffe versenken“-Spiel zu entwickeln, um Kindern mit motorischen Einschränkungen den Umgang mit der Computer-Maus zu trainieren.

Das Projekt soll webbasiert als Demoversion zunächst auf einen lokalliegenden Server installiert werden. Der Grund für die webbasierte Entwicklung ist der, dass auf den Schüler-PCs keine zusätzliche Software dafür installiert werden soll. Die Schüler-PCs sind alle standardmäßig mit einem Browser ausgestattet. Somit ist die webbasierte Umsetzung am besten geeignet. Diese Demoversion ist zu Testzwecken auf 2 Spieler im Netzwerk beschränkt und kann in Zukunft erweitert werden, um mehrere Matches gleichzeitig im Netzwerk zu ermöglichen.

Es wird eine HTML-Datei erstellt, die mit JavaScript interaktiv bedienbar wird. Auf dem Server wird eine Textdatei mit Daten der Positionen der Schiffe und Schüsse abgelegt. Diese Informationen werden mit PHP über Ajax ausgelesen und im HTML dynamisch mit JavaScript und CSS grafisch dargestellt.

Es wurden Features und Regeln festgelegt, die im Programm umgesetzt werden sollen.

Folgende Features sollen implementiert werden:

* Spieler gegen Spieler
* Online über einen Webserver nutzbar
* Statistik für Züge, Trefferquoten und verlorene Schiffe

Es wurden folgende Regeln definiert:

1. Die Spielfeldgröße beträgt 10 x 10 Felder.
2. Der anzufangende Spieler soll zufällig ausgewählt werden.
3. Die Schiffe dürfen nicht aneinanderstoßen.
4. Die Schiffe dürfen nicht über Eck gebaut sein oder Ausbuchtungen besitzen.
5. Die Schiffe dürfen auch am Rand des Spielfelds liegen.
6. Die Schiffe dürfen nicht diagonal aufgestellt werden.
7. Jeder Spieler verfügt über insgesamt zehn Schiffe (in Klammern die Größe):

* ein Schlachtschiff (5 Felder)
* zwei Kreuzer (je 4 Felder)
* drei Zerstörer (je 3 Felder)
* vier U-Boote (je 2 Felder)
  + - Worum geht es eigentlich?
    - Was soll erreicht werden?

## Projektbegründung

Text

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

## Projektschnittstellen

Das Produkt soll in der Ojde-Schule zum Einsatz kommen

* + - 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?

## Projektabgrenzung

Text

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

1. **Projektplanung**

## Projektphasen

Text

* + - 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](#_bookmark13) zeigt ein Beispiel für eine grobe Zeitplanung.

Eine detailliertere Zeitplanung ist in [Tabelle 5](#_bookmark53) in Anhang [A1](#_bookmark52) zu sehen.

|  |  |
| --- | --- |
| **Projektphase** | **Geplante Zeit** |
| Analyse | 9 h |
| Entwurf | 20 h |
| Implementierung | 30 h |
| Abnahme | 1 h |
| Einführung | 1 h |
| Dokumentation | 9 h |
| **Gesamt** | **80 h** |

**Tabelle 1: Grobe Zeitplanung**

## Abweichungen vom Projektantrag

Text

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

## Ressourcenplanung

Text

* + - 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).

## Entwicklungsprozess

Text

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

1. **Analysephase**

## Ist-Analyse

Text

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

## Wirtschaftlichkeitsanalyse

Text

* Rentiert sich das Projekt für das Unternehmen?



### Make or Buy-Entscheidung

Das Projekt soll kundengerecht für die Ojde-Schule entwickelt werden. Dabei wird Zugriff auf den Quellcode benötigt, um bei Bedarf gewünschte Anpassungen vornehmen zu können. Somit ist ein Kauf eines Standardproduktes nicht geeignet und muss individuell angepasst entwickelt werden.

* + - * Gibt es vielleicht schon ein fertiges Produkt, das alle Anforderungen des Projekts ab- deckt?
      * Wenn ja, wieso wird das Projekt trotzdem umgesetzt?

### Projektkosten

Die Kosten der Durchführung des Projektes setzen sich aus Ressourcenkosten und Personalkosten zusammen. Der Stundensatz für Personal wird auf 70 € angesetzt. Sachmittelkosten betragen 40 € für Hardware und Software für den Zeitraum der Durchführung des Projektes.

* + - * 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).

Es ergibt sich also ein Stundensatz von *7,56 EUR*. Die Durchführungszeit des Projekts beträgt 70 Stunden. Für die Nutzung von Ressourcen2 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](#_bookmark23) und sie betragen insgesamt *2.739,20 EUR*.

|  |  |  |  |
| --- | --- | --- | --- |
| **Vorgang** | **Zeit** | **Kosten / Stunde** | **Kosten** |
| Entwicklung | 80 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 €*** |

### Amortisationsdauer

Eine Amortisierung ist nicht berechenbar. Es wird lediglich durch die Entwicklung der Demoversion ein Teil der Vorarbeit geleistet. Dies hat zur Folge, dass die spätere Arbeitszeit für das finale Produkt eingespart wird.

* + - * 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:

Dadurch ergibt sich eine jährliche Einsparung von:

= 36.680 €

Die Amortisationszeit beträgt also:

## Nutzwertanalyse

Text

* 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](#_bookmark30) ([Architekturdesign](#_bookmark30)).

## Anwendungsfälle

Text

* 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](#_bookmark55).

## Qualitätsanforderungen

Text

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

## Lastenheft/Fachkonzept

Text

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

**Beispiel**

Ein Beispiel für ein Lastenheft findet sich im Anhang [A2](#_bookmark54).

1. **Entwurfsphase**

## Zielplattform

HTML, CSS, JS, PHP, XAMPP Server, PC. Siehe Projektantrag…

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

## Architekturdesign

Text

* + - 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.

**Beispiel**

Anhand der Entscheidungsmatrix in [Tabelle 3](#_bookmark31) wurde für die Implementierung der Anwendung das PHP-Framework *Symfony* ausgewählt.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Eigenschaft** | **Gewich-**  **tung** | ***Akelos*** | ***CakePHP*** | ***Symfony*** | **Eigenent- wicklung** |
| 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**

## Entwurf der Benutzeroberfläche

Text

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

**Beispiel**

Beispielentwürfe finden sich im Anhang [A7](#_bookmark63).

## Datenmodell

JSON…

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

**Beispiel**

In Anhang [A5](#_bookmark58) wird ein ERM dargestellt, welches lediglich Entitäten, Relationen und die dazu- gehörigen Kardinalitäten enthält.

## Geschäftslogik

PAP…

* + - 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](#_bookmark76) eingesehen werden.

Die EPK in Anhang [A6](#_bookmark61) zeigt den grundsätzlichen Ablauf beim Einlesen eines Moduls.

## Maßnahmen zur Qualitätssicherung

Text

* + - Welche Maßnahmen werden ergriffen, um die Qualität des Projektergebnisses (siehe Kapitel [3.5](#_bookmark26)) zu sichern (z.B. automatische Tests, Anwendertests)?
    - Ggfs. Definition von Testfällen und deren Durchführung (durch Programme/Benutzer).

## Pflichtenheft/Datenverarbeitungskonzept

Text

* + - 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](#_bookmark27)) aufbauende Pflichtenheft ist im Anhang [A4](#_bookmark57) zu finden.

1. **Implementierungsphase**

## Implementierung der Datenstrukturen

JSON

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

## Implementierung der Benutzeroberfläche

Text

* + - 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](#_bookmark66).

## Implementierung der Spiellogik

Jede ansprechbare Zelle wurde eine ID zugewiesen, die einem Muster folgt. Sie setzt sich auf Zeile und Spalte zusammen und gleicht damit den Aufbau eines Koordinatensystems. (siehe Abbildung unten)

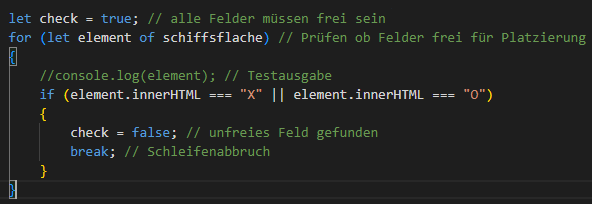


Der Ursprung ist das Feld „A1“, wobei das „a“ die Zeile beschriftet und die „1“ die Spalte. Die Beschriftung geht bis „j10“

### Platzierung der Schiffe

Nach dem das Design der Oberfläche der Webseite für das Spiel „Schiffe versenken“ fertiggestellt wurde, wurde zunächst die Platzierung der Schiffe ermöglicht. Dazu wurden die Aktionen des Spielers gespeichert, durch die ein Spieler Start- und Endkoordinaten des eigenen Spielfeldes des Schiffes durch einen Klick auf das gewünschte Spielfeld auswählen kann.

Hierbei werden die für die Platzierung benötigten Spielfeldflächen zunächst in eine Liste geschrieben. Es werden zunächst alle in der Liste enthaltenden Flächen überprüft, ob diese frei zur Platzierung des Schiffes sind. Dies ist nur der Fall, wenn keines der Flächen bereits durch ein Schiff belegt wurde oder es sich auch nicht um ein Sperrfeld handelt. (siehe Ausschnitt aus Quellcodedokumentation unten)



Frage: Codeschnipsel okay?

Eine Sperrfläche wird direkt um das Schiff nach der Platzierung erstellt, um zu verhindert, dass Schiffe aneinander platziert werden können.

### Anmeldung der Spieler

A

### Spielzüge

* + - 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](#_bookmark74).

1. **Abnahmephase**

Text

* + - 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](#_bookmark71). Dort ist auch der Aufruf des Tests auf der Konsole des Webservers zu sehen.

1. **Einführungsphase**

Text

* + - 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?

1. **Dokumentation**

Text

* + - 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 je- den Fall in einer Dokumentation für den IT-Bereich).

**Beispiel**

Ein Ausschnitt aus der erstellten Benutzerdokumentation befindet sich im Anhang [A13](#_bookmark78).

Die Entwicklerdokumentation wurde mittels *PHPDoc* automatisch generiert. Ein beispielhafter Auszug aus der Dokumentation einer Klasse findet sich im Anhang [A9](#_bookmark69).

1. **Fazit**

## Soll-/Ist-Vergleich

Text

* + - 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](#_bookmark46) zu erkennen ist, konnte die Zeitplanung bis auf wenige Ausnahmen eingehal- ten 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** | **80 h 80 h**  **Tabelle 4: Soll-/Ist-Vergleich** |  |

## Fazit

Text

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

## Ausblick

Text

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

**Literaturverzeichnis**

1. 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.

1. Anhang

## A1 Detaillierte Zeitplanung

|  |  |  |  |
| --- | --- | --- | --- |
| **Analysephase** |  |  | **9 h** |
| 1. Analyse des Ist-Zustands |  | 3 h |  |
| 1.1. Fachgespräch mit der EDV-Abteilung | 1 h |  |  |
| 1.2. Prozessanalyse | 2 h |  |  |
| 2. „Make or buy“-Entscheidung und Wirtschaftlichkeitsanalyse |  | 1 h |  |
| 3. Erstellen eines Use-Case-Diagramms |  | 2 h |  |
| 4. Erstellen des Lastenhefts mit der EDV-Abteilung |  | 3 h |  |
| **Entwurfsphase** |  |  | **20 h** |
| 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 |  |
| **Implementierungsphase** |  |  | **30 h** |
| 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 |  |

|  |  |  |
| --- | --- | --- |
| **Abnahmetest der Fachabteilung** |  | **1 h** |
| 1. Abnahmetest der Fachabteilung | 1 h |  |
| **Einführungsphase** |  | **1 h** |
| 1. Einführung/Benutzerschulung | 1 h |  |
| **Erstellen der Dokumentation** |  | **9 h** |
| 1. Erstellen der Benutzerdokumentation | 2 h |  |

|  |  |  |
| --- | --- | --- |
| 2. Erstellen der Projektdokumentation |  | 6 h |
| 3. Programmdokumentation |  | 1 h |
| 3.1. Generierung durch *PHPdoc* | 1 h |  |
| **Gesamt** |  | **80 h** |

**Tabelle 5: Detaillierte Zeitplanung**

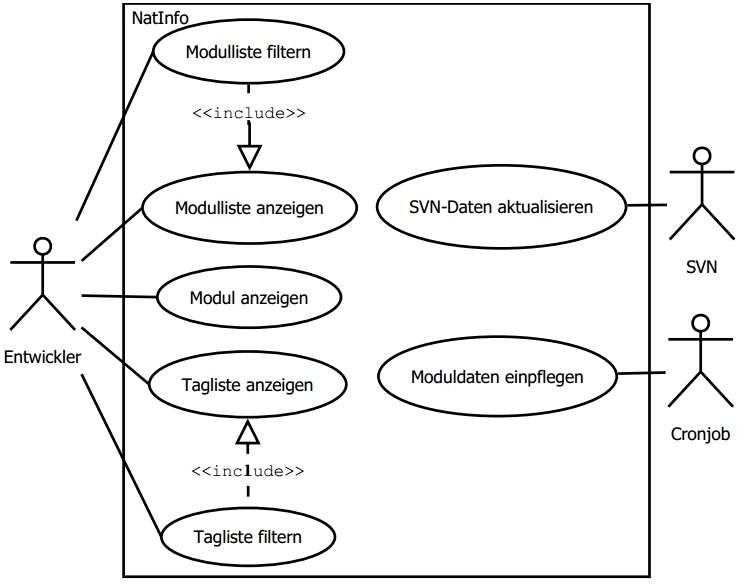
## A2 Lastenheft (Auszug)

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

**Die Anwendung muss folgende Anforderungen erfüllen.**

1. Verarbeitung der Moduldaten
   1. Die Anwendung muss die von Subversion und einem externen Programm bereitgestell- ten 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 Ein- haltung 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 Bearbei- tung 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 die- sem.
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 automa- tisch aktualisiert werden.
   3. Es muss ermittelt werden, ob Änderungen auf der Produktionsumgebung vorgenom- men 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 Entwicklungspro- zess einfach angepasst werden kann.

## A3 Use-Case-Diagramm



**Abbildung 1: Use-Case-Diagramm**

## A4 Pflichtenheft (Auszug)

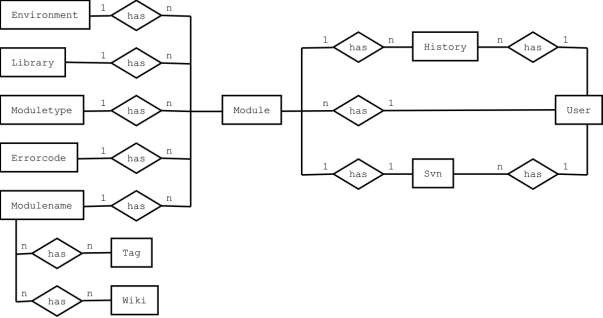
**Zielbestimmung**

1. Musskriterien
   1. Modul-Liste: Zeigt eine filterbare Liste der Module mit den dazugehörigen Kerninforma- tionen 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 Pro- zesses 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 Da- ten 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 ei- nem 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 Pro- grammtyp, 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 Da- tenbank 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, wel- ches 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 ge- setzte Tags/Wikiartikel entfernt.
   6. Sonstiges
      * Das Programm läuft als Webanwendung im Intranet.
      * Die Anwendung soll möglichst leicht erweiterbar sein und auch von anderen Entwick- lungsprozessen ausgehen können.
      * Eine Konfiguration soll möglichst in zentralen Konfigurationsdateien erfolgen.

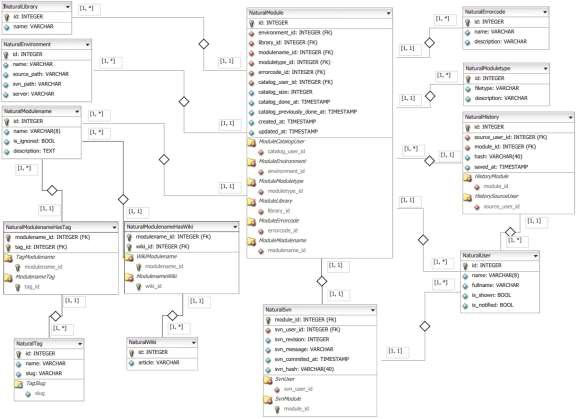
**Produkteinsatz**

1. Anwendungsbereiche
   1. Die Webanwendung dient als Anlaufstelle für die Entwicklung. Dort sind alle Informati- onen 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 Versions- verwaltung, das Wiki und der nächtliche Export sind bereits vorhanden und konfiguriert. Durch einen täglichen Cronjob werden entsprechende Daten aktualisiert, die Weban- wendung ist jederzeit aus dem Intranet heraus erreichbar.

## A5 Datenbankmodell

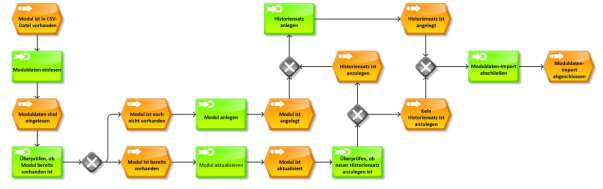


**Abbildung 2: Entity-Relationship-Model**



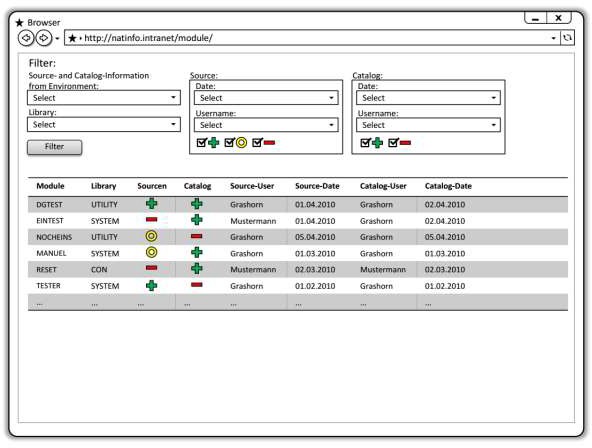
**Abbildung 3: Tabellenmodell**

## A6 Ereignisgesteuerte Prozesskette

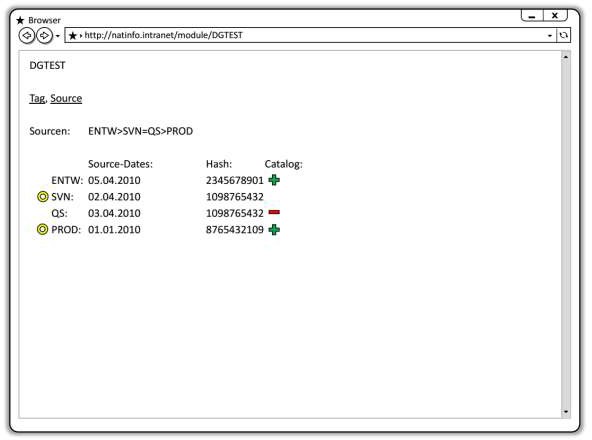


**Abbildung 4: Prozess des Einlesens eines Moduls**

## A7 Oberflächenentwürfe



**Abbildung 5: Liste der Module mit Filtermöglichkeiten**



**Abbildung 6: Anzeige der Übersichtsseite einzelner Module**

## A8 Screenshots der Anwendung

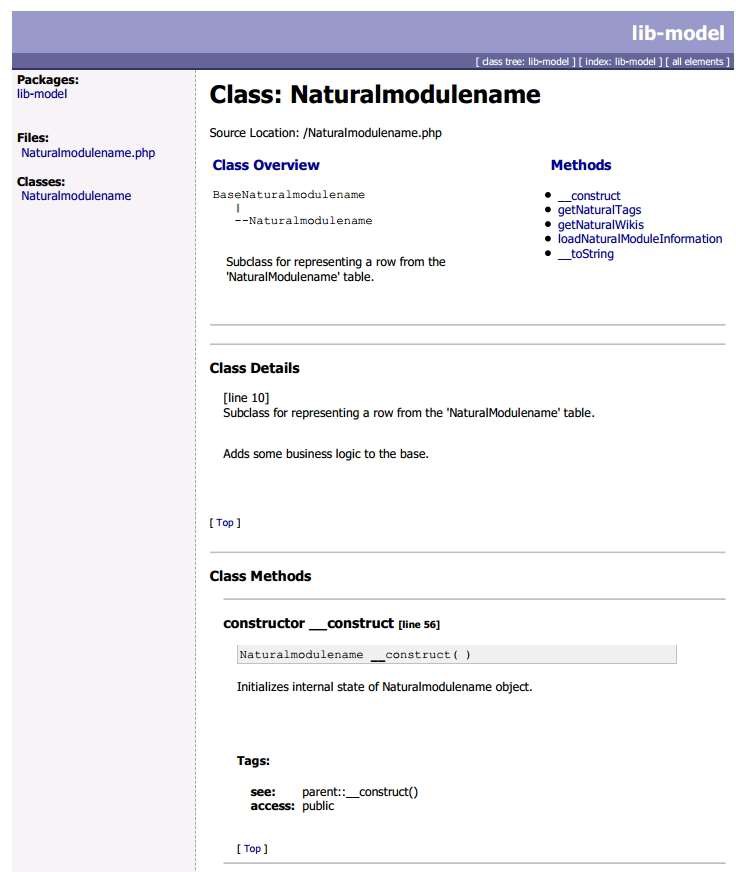


**Abbildung 7: Anzeige und Filterung der Module nach Tags**



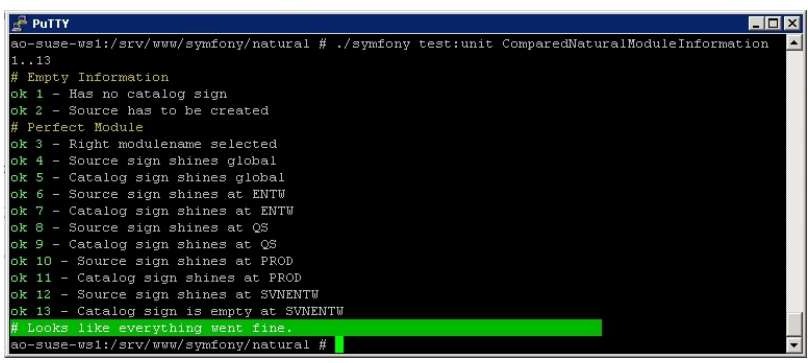
**Abbildung 8: Liste der Module mit Filtermöglichkeiten**

## 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;

private function containsCatalogSign($sign) {

foreach($this->naturalModuleInformations as $information) { if($information->getCatalogSign() == $sign) {

return true;

}

}

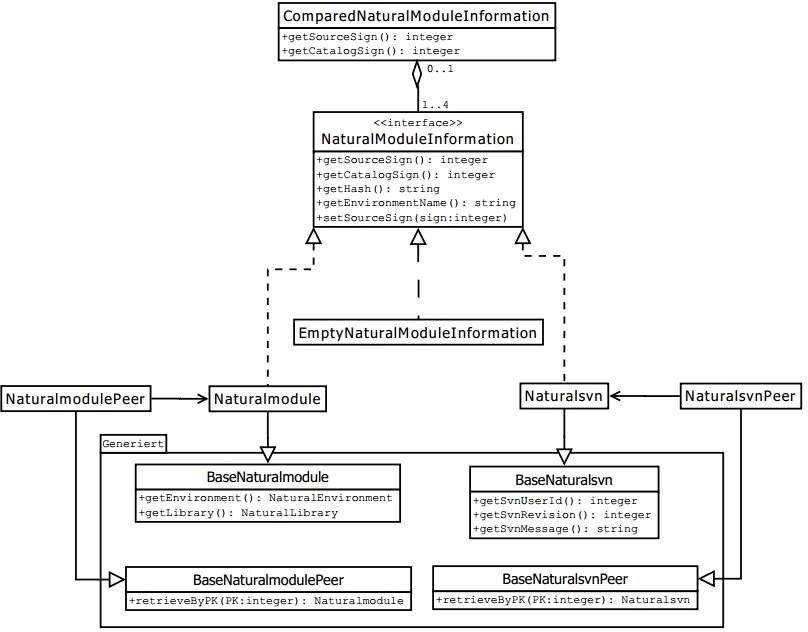
return false;

}

}

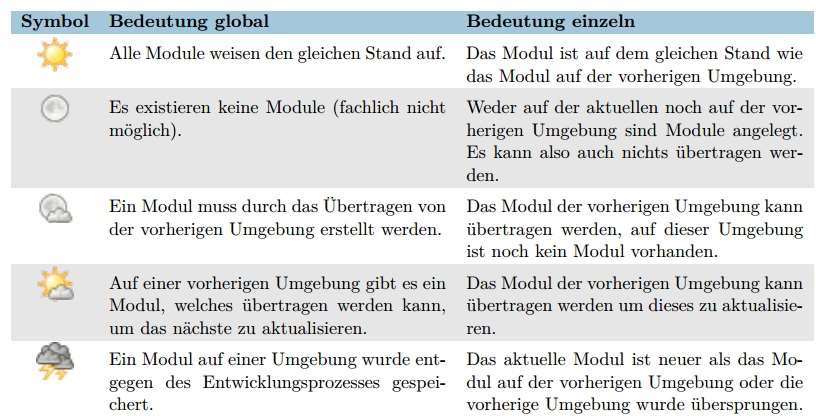
**Listing 2: Klasse ComparedNaturalModuleInformation**

## A12 Klassendiagramm



**Abbildung 11: Klassendiagramm**

## A13 Benutzerdokumentation (Auszug)



**Abbildung 12: Auszug aus der Benutzerdokumentation**