

Prüfungsteil A

Prüfling (private Anschrift):

Ausbildungsbetrieb:

Bestätigung über durchgeführte Projektarbeit

diese Bestätigung ist mit der Projektdokumentation einzureichen

Ausbildungsberuf (bitte unbedingt angeben):

Projektbezeichnung:

Projektbeginn: _____ Projektfertigstellung: _____ Zeitaufwand in Std.: _____

Bestätigung der Ausbildungsfirma:

Wir bestätigen, dass der/die Auszubildende das oben bezeichnete Projekt einschließlich der Dokumentation im Zeitraum

vom: _____ bis: _____ selbständig ausgeführt hat.

Projektverantwortliche(r) in der Firma:

Vorname	Name	Telefon	Unterschrift
---------	------	---------	--------------

Ausbildungsverantwortliche(r) in der Firma:

Vorname	Name	Telefon	Unterschrift
---------	------	---------	--------------

Eidesstattliche Erklärung:

Ich versichere, dass ich das Projekt und die dazugehörige Dokumentation selbständig erstellt habe.

Ort und Datum: _____ Unterschrift des Prüflings: _____



Abschlussprüfung Winter 2023/2024

Fachinformatiker (VO 2020) Fachrichtung: Anwendungsentwicklung
Dokumentation zur betrieblichen Projektarbeit

Workflow-Anpassungen

Anpassen von Workflows für das Änderungsmanagement in
Windchill

Abgabetermin: Pfaffenhofen, den 30.11.2023

Prüfungsbewerber:

Fabian Schmidberger

Am Galgenfeld 4a

86554 Pöttmes

Prüflingsnummer: 20525



Ausbildungsbetrieb:

Arsandis GmbH

Angkofen 5

85276 Pfaffenhofen an der Ilm

Ansprechpartner: Joachim Loos

Inhaltsverzeichnis

Abbildungsverzeichnis	III
Tabellenverzeichnis	IV
Listings	V
Abkürzungsverzeichnis	VI
1 Einleitung	1
1.1 Vorstellung der Arsandis GmbH	1
1.2 Projektbeschreibung	1
1.3 Projektumfeld	1
1.4 Projektziel	1
1.5 Projektbegründung	2
1.6 Projektschnittstellen	2
1.7 Voraussetzungen für das Verständnis	3
1.7.1 Workflows	3
1.7.2 Engineering Change Management, Änderungsmanagement	3
1.7.3 Engineering Change Request, Änderungsantrag	3
1.7.4 Engineering Change Notice, Änderungsmitteilung	3
1.7.5 Engineering Change Activity, Änderungsaufgabe	4
1.7.6 Promotion Request	4
2 Projektplanung	4
2.1 Projektphasen	4
2.2 Ressourcenplanung	5
2.3 Entwicklungsprozess	5
3 Analysephase	5
3.1 Ist-Analyse	5
3.2 Wirtschaftlichkeitsanalyse	7
3.2.1 Projektkosten	7
3.2.2 Amortisation	7
3.3 Anwendungsfälle	7
3.4 Qualitätsanforderungen	7
3.5 Lastenheft/Fachkonzept	7
4 Entwurfsphase	7
4.1 Zielplattform	7
4.2 Architekturdesign	8
4.3 Entwurf der Benutzeroberfläche	8
4.4 Datenmodell	8
4.5 Geschäftslogik	9

4.6	Maßnahmen zur Qualitätssicherung	9
4.7	Pflichtenheft/Datenverarbeitungskonzept	10
5	Implementierungsphase	10
5.1	Implementierung der Datenstrukturen	10
5.2	Implementierung der Benutzeroberfläche	10
5.3	Implementierung der Geschäftslogik	10
6	Abnahmephase	11
7	Einführungsphase	11
8	Dokumentation	11
9	Fazit	12
9.1	Soll-/Ist-Vergleich	12
9.2	Lessons Learned	12
9.3	Ausblick	12
	Eidesstattliche Erklärung	13
A	Anhang	i
A.1	Detaillierte Zeitplanung	i
A.2	Lastenheft (Auszug)	i
A.3	Use Case-Diagramm	iii
A.4	Pflichtenheft (Auszug)	iii
A.5	Datenbankmodell	v
A.6	Oberflächenentwürfe	vi
A.7	Screenshots der Anwendung	viii
A.8	Entwicklerdokumentation	x
A.9	Testfall und sein Aufruf auf der Konsole	xii
A.10	Klasse: ComparedNaturalModuleInformation	xiii
A.11	Klassendiagramm	xvi
A.12	Benutzerdokumentation	xvii

Abbildungsverzeichnis

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

Tabellenverzeichnis

1	Zeitplanung	4
2	Entscheidungsmatrix	8
3	Soll-/Ist-Vergleich	12
4	Add caption	i

Listings

1	Testfall in PHP	xii
2	Klasse: ComparedNaturalModuleInformation	xiii

Abkürzungsverzeichnis

AWS	Applied Water Systems
UML	Unified Modeling Language
XGV	Xylem Global Vault
ECR	Engineering Change Request, Änderungsantrag
ECN	Engineering Change Notice, Änderungsmitteilung
ECM	Engineering Change Management, Änderungsmanagement
ECA	Engineering Change Activity, Änderungsaufgabe
OOTB	out-of-the-box
PR	Promotion Request
VM	Virtuelle Maschine

1 Einleitung

1.1 Vorstellung der Arsandis GmbH

Die Arsandis GmbH ist ein IT- und Dienstleistungsunternehmen in Angkofen, Pfaffenhofen an der Ilm. Gegründet wurde die Arsandis im Jahr 2015 und beschäftigt zur Zeit 13 Mitarbeiter.

Das Hauptgeschäftsfeld der Arsandis GmbH liegt in der Beratung von Fertigungsunternehmen im Bereich des Product Lifecycle Managements (PLM). Darüber hinaus hat das Unternehmen ein starkes Interesse an zukunftsorientierten Technologien und verfolgt aktiv die Entwicklungen in Bereichen wie Augmented Reality, Virtual Reality und Internet of Things. Diese Technologien bieten innovative Lösungen für die Industrie der Zukunft. Weitere zukunftsorientierte Technologien werden ebenfalls verfolgt, darunter zum Beispiel AR-, VR- und IoT-basierte Lösungen für die Industrie der Zukunft. Die Arsandis GmbH hat ein breites Spektrum an Kunden aus verschiedenen Branchen. Dazu gehören Unternehmen aus dem Maschinenbau, der Automobil- und Luftfahrtindustrie sowie der HighTech- und Telekommunikationsbranche.

Als offizieller Partner von PTC liegt das Hauptaugenmerk hier meist auf dem PTC-Portfolio, das ein umfangreiches Ökosystem für die Fertigungsindustrie bereitstellt.

1.2 Projektbeschreibung

1.3 Projektumfeld

Das Projekt wird im Auftrag der amerikanischen Firma Applied Water Systems (AWS) durchgeführt, die in der Herstellung von Wasserpumpen tätig ist. AWS ist ein Tochterunternehmen von Xylem, einem Unternehmen, das sich auf die Entwicklung und Bereitstellung von Wassertechnologien spezialisiert hat. AWS nutzt den Windchill Server von Xylem, der als Xylem Global Vault (XGV) bekannt ist. Die meisten Tochterunternehmen von Xylem, einschließlich AWS, sind auf diesem Server als Organisation eingerichtet.

Durchgeführt wird das Projekt von mir in den Räumen der Arsandis GmbH. Im Vornherein wurde für dieses Projekt eine Windchill Entwicklungsumgebung auf unserem Unternehmensserver eingerichtet. Auf dieser findet jegliche Entwicklungsarbeit statt. Nach Abschluss der Implementierungsphase sollen die Softwareanpassungen in das Testsystem von Xylem integriert werden. Die finalen Tests und das Deployment auf das Produktivsystem werden von Mitarbeitern von AWS durchgeführt.

1.4 Projektziel

Ziel des Projektes ist es, das bestehende Engineering Change Management, Änderungsmanagement (ECM) von AWS anzupassen und zu erweitern. Dabei sollen die folgenden Funktionen implementiert werden:

1 Einleitung

- Übersichtlichere und besser nachvollziehbarere Darstellung der Workflowprozesse, die es einfacher macht, den Workflow in Zukunft zu erweitern.
- Hinzufügen neuer Abteilungen zum Prozess
- Die JSP-Seite zur Auswahl von Rollen soll durch eine out-of-the-box (OOTB) Lösung ersetzt werden. Dies gewährleistet, dass zukünftige Upgrades des Windchillsystems zu keinen Kompatibilitätsproblemen führen.
- Die Implementierungsaufgaben der Abteilungen aus der Engineering Change Notice, Änderungsmitteilung (ECN) sollen in eine Engineering Change Activity, Änderungsaufgabe (ECA) ausgelagert werden, um den Change Prozess nachvollziehbarer zu gestalten. Dies erlaubt es uns auch zusätzliche ECA spezifische Windchill Features zu nutzen.
- Die Logik der Startparameter soll angepasst werden, dazu muss Code im Workflow angepasst werden.
- Objekte, die gerade vom Change Admin überprüft werden, sollen für eine weitere Bearbeitung per Programmierung gesperrt werden, damit keine ungewünschten Änderungen erfolgen können.
- Workflowprozesse so anpassen, dass auch Windchill Admins ohne Zugriff zur Virtuellen Maschine des Produktivsystems Änderungen am Prozess durchführen können.
- Umbenennen von Rollen, die am Prozess beteiligt sind.

1.5 Projektbegründung

- Warum ist das Projekt sinnvoll (z. B. Kosten- oder Zeitersparnis, weniger Fehler)?
- Was ist die Motivation hinter dem Projekt?
- bessere Robustheit des Prozesses
- Zeitersparnis
- reduction in EC Cycle Time

Zum einen wird eine erhöhte Robustheit des aktuellen Prozesses angestrebt, indem das manuelle Erstellen von Promotion Requests (PRs) automatisiert wird. Zum anderen soll der Prozess durch neue Funktionen erweitert werden.

1.6 Projektschnittstellen

- System: XGV Windchill-Server
- aktuell bestehende ECM-Workflows
- keine Änderungen für andere Organisationen

1.7 Voraussetzungen für das Verständnis

1.7.1 Workflows

Workflows sind Objekte in Windchill, die es einem erlauben, Unternehmensprozesse abzubilden. Dafür stellt Windchill einen Workflow-Editor bereit, mit dem man die Unternehmensprozesse in einer praktischen Benutzeroberfläche definieren kann. Innerhalb von Workflows lässt sich auch direkt Code definieren, der dann mit dem Workflow ausgeführt wird. Hier kann man auch auf Klassen und Methoden innerhalb der Windchill-Codebase verweisen. Die folgenden Konzepte wurden und werden mit Hilfe von Workflows umgesetzt.

1.7.2 Engineering Change Management, Änderungsmanagement

Das Änderungsmanagement ist ein essenzieller Bestandteil von Windchill. Teile, wie z.B. CAD-Dateien oder Dokumente kommen in ihrem Lebenszyklus zumeist irgendwann an den Punkt, an dem sie für die Produktion freigegeben werden. Da bereits umfassende Ressourcen für die Produktion aufgewendet wurden, muss eine nachträgliche Änderung gut begründet werden. Schließlich wurden bereits Maschinen/Maschinenteile bestellt, eingerichtet oder modifiziert, um das Produkt herzustellen. Demzufolge müssen nachträgliche Änderungen nun in Absprache mit verschiedenen Unternehmensabteilungen koordiniert werden. Der Change Admin ist für die Koordination des Change Managements verantwortlich. In Windchill gibt es eigentlich drei verschiedene Change Administrator Rollen, vereinfacht wird folgend aber lediglich vom Change Admin gesprochen.

Hier kommt das Änderungsmanagement ins Spiel. Es umfasst insgesamt drei Bestandteile, die alle in Windchill implementiert sind: Engineering Change Request, Änderungsantrag, Engineering Change Notice, Änderungsmitteilung und Engineering Change Activity, Änderungsaufgabe. Diese werden in den nächsten Abschnitten genauer erläutert.

1.7.3 Engineering Change Request, Änderungsantrag

Der Änderungsantrag ist der erste Schritt im Änderungsmanagement. Grob gesagt wird im Engineering Change Request, Änderungsantrag (ECR) geklärt, ob die Änderung umgesetzt werden kann. Der Change Admin wählt die Unternehmensabteilungen aus, die über die Durchführung der Änderung abstimmen sollen. Die Abteilungen stimmen dann entweder für oder gegen die geplante Änderung. Nur wenn alle befragten Abteilungen für die Änderung sind, wird der Prozess mit der ECN fortgesetzt. Stimmt mindestens eine Abteilung dagegen, so wird der Änderungsantrag verworfen.

1.7.4 Engineering Change Notice, Änderungsmitteilung

Die Änderungsmitteilung ist der zweite Schritt im Änderungsmanagement. Hier geht es darum, dass festgelegt wird, wie die Änderung umgesetzt wird. Dafür legt der Change-Admin zuerst fest,

welche Abteilungen an der Umsetzung der Änderungen beteiligt sein sollen. Danach bestimmt er, welche Nutzer aus der Abteilung an der Umsetzung der Änderung arbeiten sollen.

1.7.5 Engineering Change Activity, Änderungsaufgabe

Die Änderungsaufgabe ist der dritte und letzte Schritt im Änderungsmanagement. Die geplanten Änderungen werden hier umgesetzt und anschließend vom Change-Admin überprüft. Ist der Change-Admin unzufrieden mit den umgesetzten Änderungen, so kann er eine Überarbeitung von den relevanten Abteilungen anfordern. Sobald der Change-Administrator seine Zustimmung zu den Änderungen erteilt hat, markiert dies den erfolgreichen Abschluss der Änderungsaufgabe und damit auch des gesamten Change-Prozesses.

1.7.6 Promotion Request

Der Erhöhungsantrag ist nicht direkt Teil dieses Projekts. Allerdings werden neue Mechanismen im ECM-Workflow eingeführt, die das manuelle Erstellen von Erhöhungsanträgen ersetzen soll. Demzufolge wird der Erhöhungsantrag kurz erläutert.

Ein Windchill-Objekt durchläuft stets verschiedene Phasen seines Lebenszyklus. Im Moment der Erstellung befindet es sich üblicherweise im Status 'In Arbeit'. Um das Objekt in einen anderen Zustand zu überführen, wie beispielsweise 'Freigegeben', wird ein Erhöhungsantrag durchlaufen. In diesem Prozess entscheidet der Genehmiger, ob eine Änderung des Status des Objekts angebracht ist.

2 Projektplanung

2.1 Projektphasen

Für die Umsetzung des Projekts wurden 76 Stunden angesetzt. Der Start des Projekts erfolgt am 07.11.2023 und bis zum 27.11.2023 wird es abgeschlossen sein. Der Tabelle 1 kann entnommen werden, in welche Hauptphasen ich das Projekt gegliedert habe.

Projektphase	Geplante Zeit
Analysephase	12 h
Entwurfsphase	3 h
Implementierungsphase inkl. Tests	45 h
Abnahme und Einführung	11 h
Erstellen der Dokumentation	5 h
Gesamt	76 h

Tabelle 1: Zeitplanung

Eine detailliertere Zeitplanung findet sich im Anhang A.1: Detaillierte Zeitplanung auf Seite i.

2.2 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).

Für die Durchführung des Projekts wurden folgende Soft- und Hardwareressourcen verwendet. Von meinem Büroarbeitsplatz aus wird mein Arbeitslaptop genutzt. Im Vorfeld des Projekts wurde bereits eine virtuelle Maschine mit dem Betriebssystem Windows Server 2019 aufgesetzt. Diese ist mit 16 GB RAM ausgestattet. Auf dieser wurde Windchill in der Version 11.1 M020-CPS026 installiert. Windchill benötigt eine Oracle Datenbank der Version 19C. Windchill benötigt eine Java Runtime Environment der Version 1.8.0 Update 202. Für die Erstellung des Codes verwendet ich die Java IDE Eclipse auf der VM. Die User des Windchillsystems werden im Open Source LDAP Programm OpenDJ 3.0.0 verwaltet. Zur Erstellung der Projektdokumentation verwende ich IntelliJ mit dem TeXiFy Plugin, dieses nutzt die MiKTeX Distribution für LaTeX.

Außerdem haben mich folgende Personen bei meinem Projekt unterstützt. Der Product Owner von AWS legt die Anforderungen fest und nimmt das Projekt ab. Ein Experte für das Windchillsystem von AWS hilft mir dabei, mich in die Customizations von AWS einzuarbeiten. Als Entwickler führt der Autor die Umsetzung des Projektes durch. Der Projektleiter von Arsandis überprüft die Umsetzung und den Code. Eine PLM Beraterin von Arsandis unterstützt den Autor beim Implementieren.

2.3 Entwicklungsprozess

Die Umsetzung meines Projekts wird grundsätzlich agil ablaufen, in dem der aktuelle Status und Änderungen an den Anforderungen nach Bedarf diskutiert werden.

3 Analysephase

3.1 Ist-Analyse

Dieser Abschnitt erhält Auszüge aus dem Projektantrag, um eine Übersicht über die aktuelle Situation des Systems zu geben.

Der ECM besteht aus den zwei Komponenten ECR und ECN. Aktuell werden Aufgaben im ECN getrackt, dies ist so von Windchill eigentlich nicht vorgesehen. In Zukunft sollen Aufgaben deswegen in einem ECA abgebildet werden. Dies führt dazu, dass die konkreten Implementierungsaufgaben logisch von der Planungsphase (ECN) abgegrenzt wird, was es dem Change Admin einfacher macht, einen Überblick über die aktuelle Situation zu bekommen.

Im ECM von AWS sollen neue Abteilungen berücksichtigt werden. Dafür müssen diese dem bestehenden ECR und ECN hinzugefügt werden.

Bei der Erstellung der ECRs werden zudem Startparameter abgefragt. Das sind Boolean-Attribute, die die automatische Auswahl von Abteilungen beeinflussen. Dies ist eine Erleichterung für den Change Admin, da dann nicht jedes Mal alle Abteilungen manuell ausgefüllt werden müssen. Der ECR ist für alle Organisationen gleich. Allerdings benötigen nicht alle Organisationen alle Attribute, die auf dem ECR definiert wurden. Dafür wurde von einem anderen Softwareunternehmen eine XML-Datei entwickelt, die nicht benötigte Attribute von einer Organisation vom UI versteckt. Es sollen neue Boolean-Attribute definiert werden, die nur für die AWS Organisationen sichtbar sind. Dazu müssen diese dem ECR hinzugefügt werden und die in der XML-Datei für die anderen Organisationen versteckt werden.

Um den Change Admin bei der Auswahl der beteiligten Abteilungen zu unterstützen, wurde von einem anderen Softwareunternehmen eine benutzerdefinierte JSP Seite entwickelt, die die Abteilungen in einer Liste darstellt. Der Change Admin kann dann über Checkboxen Abteilungen für den Prozess entfernen oder hinzufügen. Dieses Pop-Up-Fenster wurde von einem anderen Softwareunternehmen in einem bereits abgeschlossenem Projekt entwickelt.

Zudem werden Abteilungen aktuell in einem eigens dafür angefertigten Pop-Up-Fenster ausgewählt. Hier kann der Change Admin über Checkboxen Abteilungen auswählen, die am Change Prozess beteiligt sein sollen. Dieses Pop-Up-Fenster wurde von einem anderen Softwareunternehmen in einem bereits abgeschlossenem Projekt entwickelt. Allerdings bringt diese Lösung einige Probleme mit sich. Zum einen verzögern Customizations, die von den OOTB-Windchill-Konfigurationen abweichen, oft größere Windchill Upgrades, da nicht garantiert ist, dass die Customization auch in zukünftigen Versionen funktioniert. Zum anderen können nur Administratoren mit Zugriff zur virtuellen Maschine Änderungen an der Struktur des Pop-Up-Fensters vornehmen (z. B. hinzufügen oder entfernen von Abteilungen), da die Customization als JSP-Datei auf dem Server liegt. Um diese beiden Probleme zu beseitigen, soll hier auf eine Lösung zurückgegriffen werden, die bereits von Haus aus in Windchill implementiert ist, nämlich den 'Set Up Participants'-Tab. Da es sich um eine OOTB-Funktion von Windchill handelt, ist es unwahrscheinlich, dass es damit Probleme bei einem Upgrade gibt. Außerdem können Windchill-Administratoren über den Workflow Template Editor Änderungen an den Abteilungen vornehmen, ohne Zugriff auf die VM zu benötigen.

Zudem sind die aktuellen Workflowprozesse sehr unstrukturiert dargestellt und sind kaum dokumentiert. Dies macht es schwieriger für Entwickler, die nicht mit den Prozessen vertraut sind, Erweiterungen oder Änderungen vorzunehmen.

3.2 Wirtschaftlichkeitsanalyse

3.2.1 Projektkosten

3.2.2 Amortisation

3.3 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 A.3: Use Case-Diagramm auf Seite iii.

3.4 Qualitätsanforderungen

- Welche Qualitätsanforderungen werden an die Anwendung gestellt (z. B. hinsichtlich Performance, Usability, Effizienz etc.)?

3.5 Lastenheft/Fachkonzept

Zu Beginn des Projekts hat der Kunde bereits einen Entwurf des Lastenhefts ausgearbeitet, welcher zusammen mit dem Autor und dem Projektleiter von Arsandis weiter angepasst wurde. Dort sind alle Anforderungen des Auftraggebers für die Anpassungen der aktuellen Anwendung enthalten. Im Anhang A.2: Lastenheft (Auszug) auf Seite i befindet sich ein Auszug aus dem Lastenheft.

4 Entwurfsphase

4.1 Zielplattform

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

4.2 Datenmodell

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

Beispiel In Abbildung 1 wird ein ERM dargestellt, welches lediglich Entitäten, Relationen und die dazugehörigen Kardinalitäten enthält.

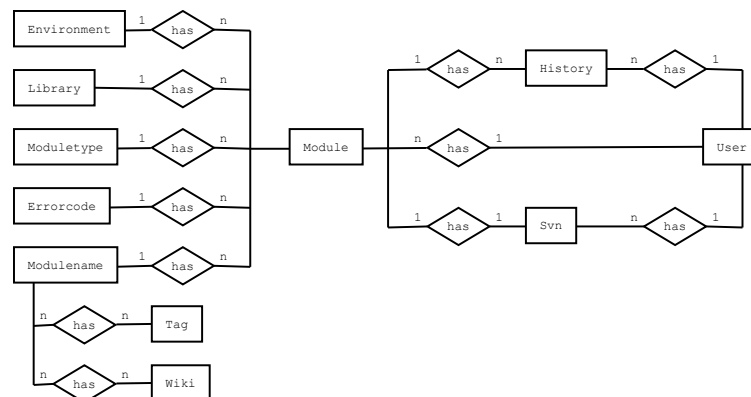


Abbildung 1: Vereinfachtes ER-Modell

4.3 Geschäftslogik

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

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

Abbildung 2 zeigt den grundsätzlichen Programmablauf beim Einlesen eines Moduls als EPK.

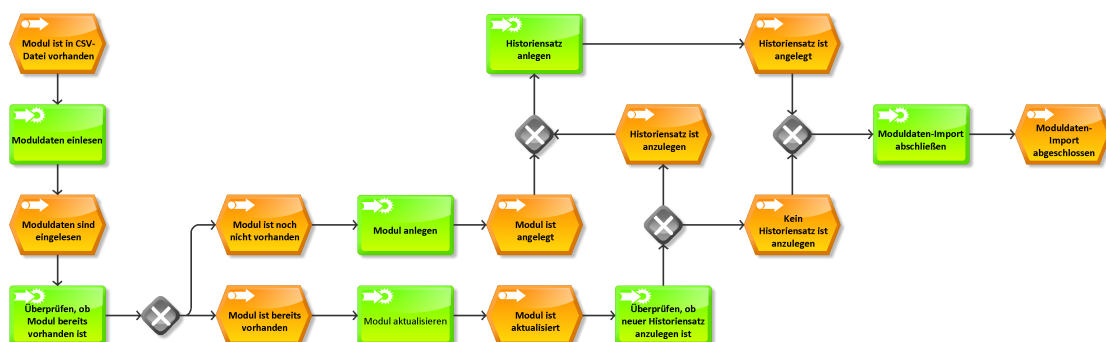


Abbildung 2: Prozess des Einlesens eines Moduls

4.4 Maßnahmen zur Qualitätssicherung

- Welche Maßnahmen werden ergriffen, um die Qualität des Projektergebnisses (siehe Kapitel 3.4: Qualitätsanforderungen) zu sichern (z. B. automatische Tests, Anwendertests)?

- Ggfs. Definition von Testfällen und deren Durchführung (durch Programme/Benutzer).

Das Testing erfolgt manuell durch “durchklicken” der Workflowprozesse und wird in zwei Phasen aufgeteilt. Zuerst erfolgt ein Arsandis-Interner Test auf unserer Entwicklungsumgebung, bei dem mich eine Kollegin unterstützt. Sobald wir unseren Testprozess erfolgreich abgeschlossen haben, werden die Anpassungen auf das Testsystems des Kunden übertragen. Daran anschließend startet die zweite Testphase, die von einigen Mitarbeitern von AWS durchgeführt werden. Nach ihren Tests erhalten wir ein Testprotokoll, in dem alle Fehler beziehungsweise Auffälligkeiten dokumentiert wurden.

Ein großer Teil des Codes wird direkt in den Workflows geschrieben, wodurch sich der Einsatz von Unit-Tests als sehr schwierig bis unmöglich gestaltet, weswegen es keine automatisierten Tests gibt.

4.5 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 3.5: Lastenheft/Fachkonzept) aufbauende Pflichtenheft ist im Anhang A.4: Pflichtenheft (Auszug) auf Seite iii zu finden.

5 Implementierungsphase

- Anpassung der Workflow-Templates
- Anlegen neuer Types

OIR

Lifecycle Template

- Anpassen der Lifecycle Templates
- Erstellung der Java-Klassen

Beispiel Die Klasse `ComparedNaturalModuleInformation` findet sich im Anhang A.10: Klasse: `ComparedNaturalModuleInformation` auf Seite xiii.

6 Abnahmephase

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

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

7 Einführungsphase

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

8 Dokumentation

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

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

¹Vgl. ?

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

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

Tabelle 2: 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)?

Eidesstattliche Erklärung

Ich, Fabian Schmidberger, versichere hiermit, dass ich meine **Dokumentation zur betrieblichen Projektarbeit** mit dem Thema

Workflow-Anpassungen – Anpassen von Workflows für das Änderungsmanagement in Windchill

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.

Pfaffenhofen, den 30.11.2023

FABIAN SCHMIDBERGER

A Anhang

A.1 Detaillierte Zeitplanung

Tabelle 3: Add caption

Abnahme und Einführung	11
Abnahme durch den Kunden	2
Deployment der Anwendung auf den Testserver	5
Review durch den Kunden	2
Review durch den Projektbetreuer	2
Analysephase	12
Analyse der bestehenden JSP-Seiten und XML-Dateien zum Customizing	3
Durchführung der Ist-Analyse	8
Erstellung eines Anwendungsfall-Diagramms und Ermittlung der Anwendungsfälle	1
Dokumentation	5
Erstellung des Benutzerhandbuchs	5
Entwurf	3
Erstellen von Aktivitätsdiagrammen zur Veranschaulichung der Workflowprozesse	3
Implementierung inkl. Tests	45
Anpassungen an der Programmcode-Logik der Startparameter	1
Einrichtung des Windchill-Entwicklungssystems (Import der Kundenkonfiguration)	2
Ersetzen der benutzerdefinierten Seite mit Windchill out-of-the-box Lösungen	2
Erweiterung der bestehenden JSP-Seiten und XML-Dateien zum Customizing	1
Hinzufügen von neuen Abteilungen zum Change Management Prozess	1
Integration der Change Activity in den bestehenden Change Management Prozess	4
Programmatisches Sperren von Objekten, die sich gerade in der Überprüfungsphase befinden	10
Programmatisches Umbenennen von Rollen	5
Testen des kompletten Change Management Prozesses	5
Workflowprozesse so gestalten, dass eine Ergänzung von neuen Abteilungen einfacher funktioniert	4
Workflowprozesse überarbeiten (übersichtlicher gestalten, Code vereinfachen)	10
Summe	76

A.2 Lastenheft (Auszug)

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

Die Anwendung muss folgende Anforderungen erfüllen:

1. Verarbeitung der Moduldaten

- 1.1. Die Anwendung muss die von Subversion und einem externen Programm bereitgestellten Informationen (z.B. Source-Benutzer, -Datum, Hash) verarbeiten.
- 1.2. Auslesen der Beschreibung und der Stichwörter aus dem Sourcecode.

2. Darstellung der Daten

- 2.1. Die Anwendung muss eine Liste aller Module erzeugen inkl. Source-Benutzer und -Datum, letztem Commit-Benutzer und -Datum für alle drei Umgebungen.
- 2.2. Verknüpfen der Module mit externen Tools wie z.B. Wiki-Einträgen zu den Modulen oder dem Sourcecode in Subversion.
- 2.3. Die Sourcen der Umgebungen müssen verglichen und eine schnelle Übersicht zur Einhaltung des allgemeinen Entwicklungsprozesses gegeben werden.
- 2.4. Dieser Vergleich muss auf die von einem bestimmten Benutzer bearbeiteten Module eingeschränkt werden können.
- 2.5. Die Anwendung muss in dieser Liste auch Module anzeigen, die nach einer Bearbeitung durch den gesuchten Benutzer durch jemand anderen bearbeitet wurden.
- 2.6. Abweichungen sollen kenntlich gemacht werden.
- 2.7. Anzeigen einer Übersichtsseite für ein Modul mit allen relevanten Informationen zu diesem.

3. Sonstige Anforderungen

- 3.1. Die Anwendung muss ohne das Installieren einer zusätzlichen Software über einen Webbrowser im Intranet erreichbar sein.
- 3.2. Die Daten der Anwendung müssen jede Nacht bzw. nach jedem SVN-Commit automatisch aktualisiert werden.
- 3.3. Es muss ermittelt werden, ob Änderungen auf der Produktionsumgebung vorgenommen wurden, die nicht von einer anderen Umgebung kopiert wurden. Diese Modulliste soll als Mahnung per E-Mail an alle Entwickler geschickt werden (Peer Pressure).
- 3.4. Die Anwendung soll jederzeit erreichbar sein.
- 3.5. Da sich die Entwickler auf die Anwendung verlassen, muss diese korrekte Daten liefern und darf keinen Interpretationsspielraum lassen.
- 3.6. Die Anwendung muss so flexibel sein, dass sie bei Änderungen im Entwicklungsprozess einfach angepasst werden kann.

A.3 Use Case-Diagramm

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

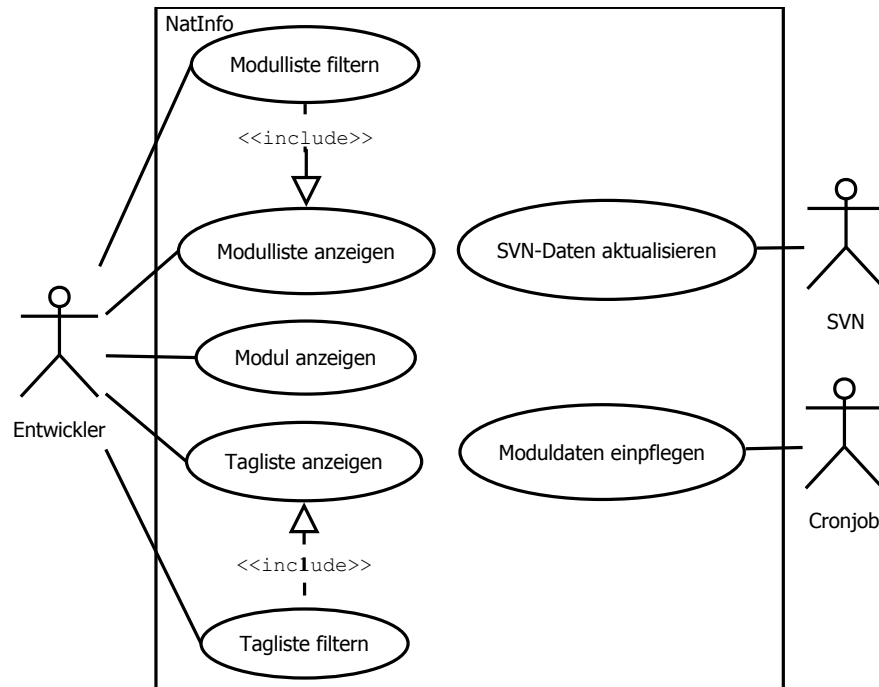


Abbildung 3: Use Case-Diagramm

A.4 Pflichtenheft (Auszug)

Zielbestimmung

1. Musskriterien

1.1. Modul-Liste: Zeigt eine filterbare Liste der Module mit den dazugehörigen Kerninformationen sowie Symbolen zur Einhaltung des Entwicklungsprozesses an

- In der Liste wird der Name, die Bibliothek und Daten zum Source und Kompilat eines Moduls angezeigt.
- Ebenfalls wird der Status des Moduls hinsichtlich Source und Kompilat angezeigt. Dazu gibt es unterschiedliche Status-Zeichen, welche symbolisieren in wie weit der Entwicklungsprozess eingehalten wurde bzw. welche Schritte als nächstes getan werden müssen. So gibt es z. B. Zeichen für das Einhalten oder Verletzen des Prozesses oder den Hinweis auf den nächsten zu tätigen Schritt.
- Weiterhin werden die Benutzer und Zeitpunkte der aktuellen Version der Sourcen und Kompilate angezeigt. Dazu kann vorher ausgewählt werden, von welcher Umgebung diese Daten gelesen werden sollen.

- Es kann eine Filterung nach allen angezeigten Daten vorgenommen werden. Die Daten zu den Sourcen sind historisiert. Durch die Filterung ist es möglich, auch Module zu finden, die in der Zwischenzeit schon von einem anderen Benutzer editiert wurden.

1.2. Tag-Liste: Bietet die Möglichkeit die Module anhand von Tags zu filtern.

- Es sollen die Tags angezeigt werden, nach denen bereits gefiltert wird und die, die noch der Filterung hinzugefügt werden könnten, ohne dass die Ergebnisliste leer wird.
- Zusätzlich sollen die Module angezeigt werden, die den Filterkriterien entsprechen. Sollten die Filterkriterien leer sein, werden nur die Module angezeigt, welche mit einem Tag versehen sind.

1.3. Import der Moduldaten aus einer bereitgestellten CSV-Datei

- Es wird täglich eine Datei mit den Daten der aktuellen Module erstellt. Diese Datei wird (durch einen Cronjob) automatisch nachts importiert.
- Dabei wird für jedes importierte Modul ein Zeitstempel aktualisiert, damit festgestellt werden kann, wenn ein Modul gelöscht wurde.
- Die Datei enthält die Namen der Umgebung, der Bibliothek und des Moduls, den Programmtyp, den Benutzer und Zeitpunkt des Sourcecodes sowie des Kompilats und den Hash des Sourcecodes.
- Sollte sich ein Modul verändert haben, werden die entsprechenden Daten in der Datenbank aktualisiert. Die Veränderungen am Source werden dabei aber nicht ersetzt, sondern historisiert.

1.4. Import der Informationen aus SVN. Durch einen „post-commit-hook“ wird nach jedem Einchecken eines Moduls ein PHP-Script auf der Konsole aufgerufen, welches die Informationen, die vom SVN-Kommandozeilentool geliefert werden, an NatInfo übergibt.

1.5. Parsen der Sourcen

- Die Sourcen der Entwicklungsumgebung werden nach Tags, Links zu Artikeln im Wiki und Programmbeschreibungen durchsucht.
- Diese Daten werden dann entsprechend angelegt, aktualisiert oder nicht mehr gesetzte Tags/Wikiartikel entfernt.

1.6. Sonstiges

- Das Programm läuft als Webanwendung im Intranet.
- Die Anwendung soll möglichst leicht erweiterbar sein und auch von anderen Entwicklungsprozessen ausgehen können.
- Eine Konfiguration soll möglichst in zentralen Konfigurationsdateien erfolgen.

Produkteinsatz

1. Anwendungsbereiche

Die Webanwendung dient als Anlaufstelle für die Entwicklung. Dort sind alle Informationen

2. Zielgruppen

3. Betriebsbedingungen

Die nötigen Betriebsbedingungen, also der Webserver, die Datenbank, die Versionsverwaltung, das Wiki und der nächtliche Export sind bereits vorhanden und konfiguriert. Durch einen täglichen Cronjob werden entsprechende Daten aktualisiert, die Webanwendung ist jederzeit aus dem Intranet heraus erreichbar.

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

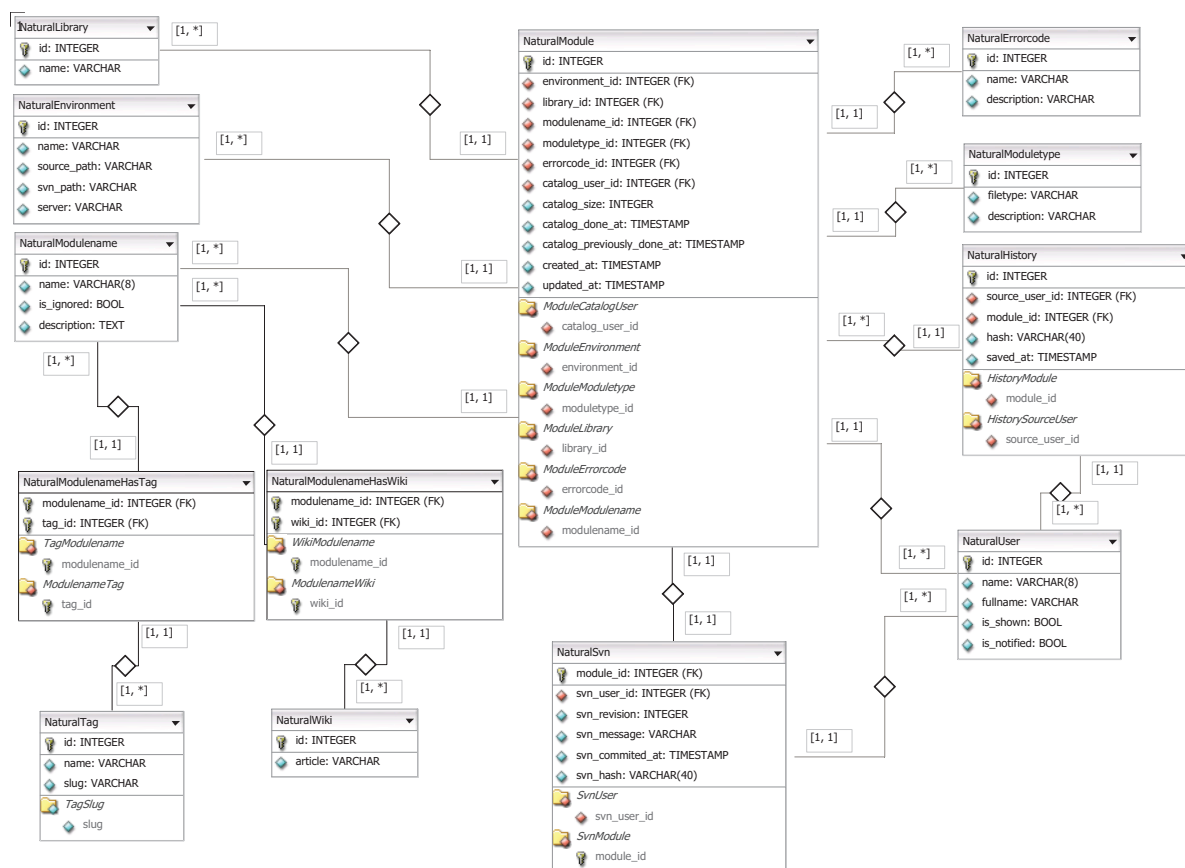


Abbildung 4: Datenbankmodell

A.6 Oberflächenentwürfe

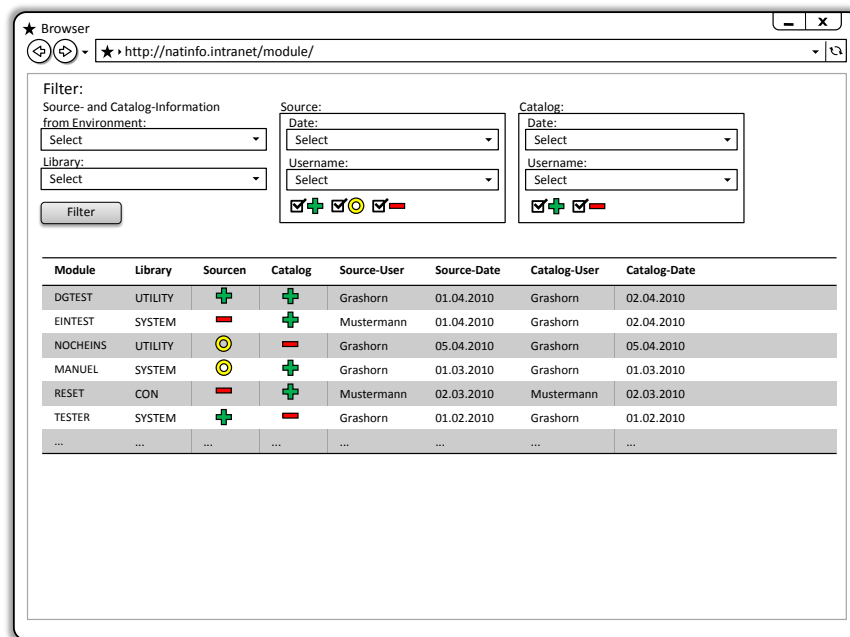


Abbildung 5: Liste der Module mit Filtermöglichkeiten

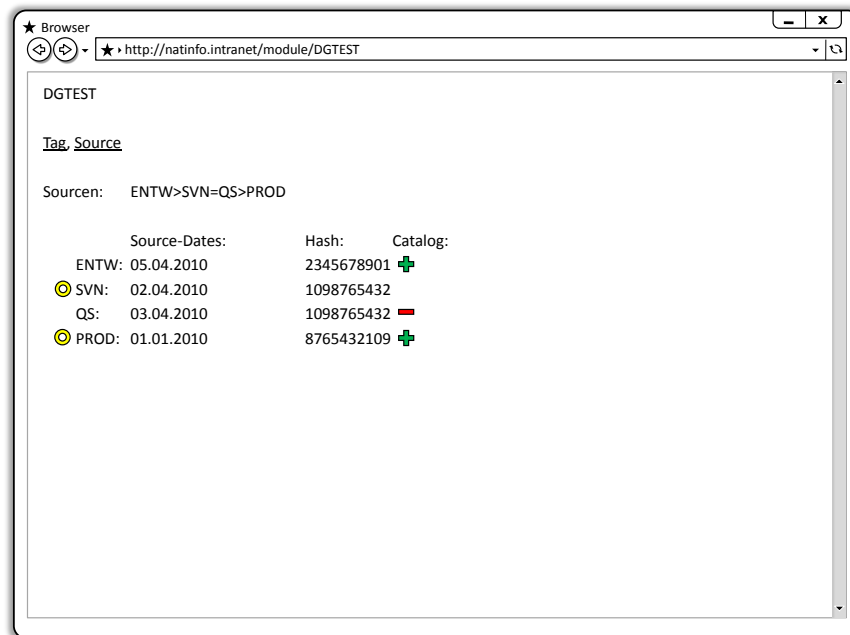


Abbildung 6: Anzeige der Übersichtsseite einzelner Module

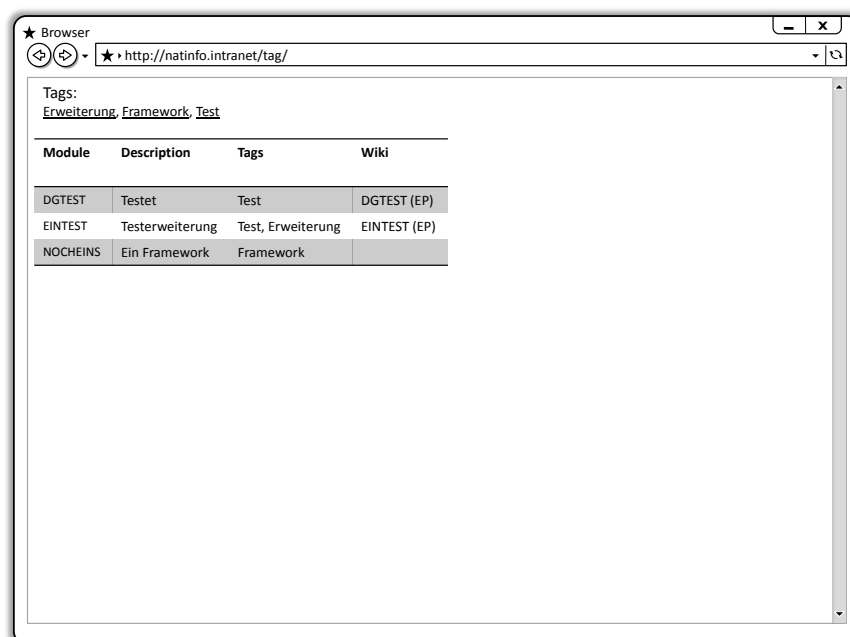


Abbildung 7: Anzeige und Filterung der Module nach Tags

A.7 Screenshots der Anwendung




Tags

Project, Test

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

Abbildung 8: Anzeige und Filterung der Module nach Tags




Häufig benötigte Seiten

[Ein-Klick-Menü](#)

Direktaufruf

[Inhaltssuche](#)

 / Modules / ENTW
Tags, Modules

Modules

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











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

Abbildung 9: Liste der Module mit Filtermöglichkeiten

A.8 Entwicklerdokumentation

lib-model

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

Packages:
[lib-model](#)

Files:
[Naturalmodulename.php](#)

Classes:
[Naturalmodulename](#)

Class: Naturalmodulename

Source Location: /Naturalmodulename.php

Class Overview

```
BaseNaturalmodulename
|
--Naturalmodulename
```

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

Methods

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

Class Details

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

Adds some business logic to the base.

[\[Top \]](#)

Class Methods

constructor [__construct](#) [line 56]

```
Naturalmodulename __construct( )
```

Initializes internal state of Naturalmodulename object.

Tags:

see: parent::__construct()
access: public

[\[Top \]](#)

method [getNaturalTags](#) [line 68]

```
array getNaturalTags( )
```

Returns an Array of NaturalTags connected with this Modulename.

Tags:

return: Array of NaturalTags
access: public

[\[Top \]](#)

method getNaturalWikis [line 83]

```
array getNaturalWikis ( )
```

Returns an Array of NaturalWikis connected with this Modulename.

Tags:

return: Array of NaturalWikis
access: public

[\[Top \]](#)

method loadNaturalModuleInformation [line 17]

```
ComparedNaturalModuleInformation  
loadNaturalModuleInformation ( )
```

Gets the ComparedNaturalModuleInformation for this NaturalModulename.

Tags:

access: public

[\[Top \]](#)

method __toString [line 47]

```
string __toString ( )
```

Returns the name of this NaturalModulename.

Tags:

access: public

[\[Top \]](#)

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


A.9 Testfall und sein Aufruf auf der Konsole

```

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

```

Listing 1: Testfall in PHP



```

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

```

Abbildung 10: Aufruf des Testfalls auf der Konsole

A.10 Klasse: ComparedNaturalModuleInformation

Kommentare und simple Getter/Setter werden nicht angezeigt.

```

1 <?php
2 class ComparedNaturalModuleInformation
3 {
4     const EMPTY_SIGN = 0;
5     const SIGN_OK = 1;
6     const SIGN_NEXT_STEP = 2;
7     const SIGN_CREATE = 3;
8     const SIGN_CREATE_AND_NEXT_STEP = 4;
9     const SIGN_ERROR = 5;
10
11     private $naturalModuleInformations = array();
12
13     public static function environments()
14     {
15         return array("ENTW", "SVNENTW", "QS", "PROD");
16     }
17
18     public static function signOrder()
19     {
20         return array(self::SIGN_ERROR, self::SIGN_NEXT_STEP, self::SIGN_CREATE_AND_NEXT_STEP,
21                     self::SIGN_CREATE, self::SIGN_OK);
22     }
23
24     public function __construct(array $naturalInformations)
25     {
26         $this->allocateModulesToEnvironments($naturalInformations);
27         $this->allocateEmptyModulesToMissingEnvironments();
28         $this->determineSourceSignsForAllEnvironments();
29     }

```

```

29
30 private function allocateModulesToEnvironments(array $naturalInformations)
31 {
32     foreach ($naturalInformations as $naturalInformation)
33     {
34         $env = $naturalInformation->getEnvironmentName();
35         if (in_array($env, self::environments()))
36         {
37             $this->naturalModuleInformations[array_search($env, self::environments())] = $naturalInformation;
38         }
39     }
40 }
41
42 private function allocateEmptyModulesToMissingEnvironments()
43 {
44     if (array_key_exists(0, $this->naturalModuleInformations))
45     {
46         $this->naturalModuleInformations[0]->setSourceSign(self::SIGN_OK);
47     }
48
49     for ($i = 0; $i < count(self::environments()); $i++)
50     {
51         if (!array_key_exists($i, $this->naturalModuleInformations))
52         {
53             $environments = self::environments();
54             $this->naturalModuleInformations[$i] = new EmptyNaturalModuleInformation($environments[$i]);
55             $this->naturalModuleInformations[$i]->setSourceSign(self::SIGN_CREATE);
56         }
57     }
58 }
59
60 public function determineSourceSignsForAllEnvironments()
61 {
62     for ($i = 1; $i < count(self::environments()); $i++)
63     {
64         $currentInformation = $this->naturalModuleInformations[$i];
65         $previousInformation = $this->naturalModuleInformations[$i - 1];
66         if ($currentInformation->getSourceSign() <> self::SIGN_CREATE)
67         {
68             if ($previousInformation->getSourceSign() <> self::SIGN_CREATE)
69             {
70                 if ($currentInformation->getHash() <> $previousInformation->getHash())
71                 {
72                     if ($currentInformation->getSourceDate('YmdHis') > $previousInformation->getSourceDate('
73                         YmdHis'))
74                     {
75                         $currentInformation->setSourceSign(self::SIGN_ERROR);
76                     }
77                     else
78                     {
79                         $currentInformation->setSourceSign(self::SIGN_NEXT_STEP);
80                     }
81                 }
82             }
83         }
84     }
85 }

```

```

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

```

Listing 2: Klasse: ComparedNaturalModuleInformation

A.11 Klassendiagramm

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

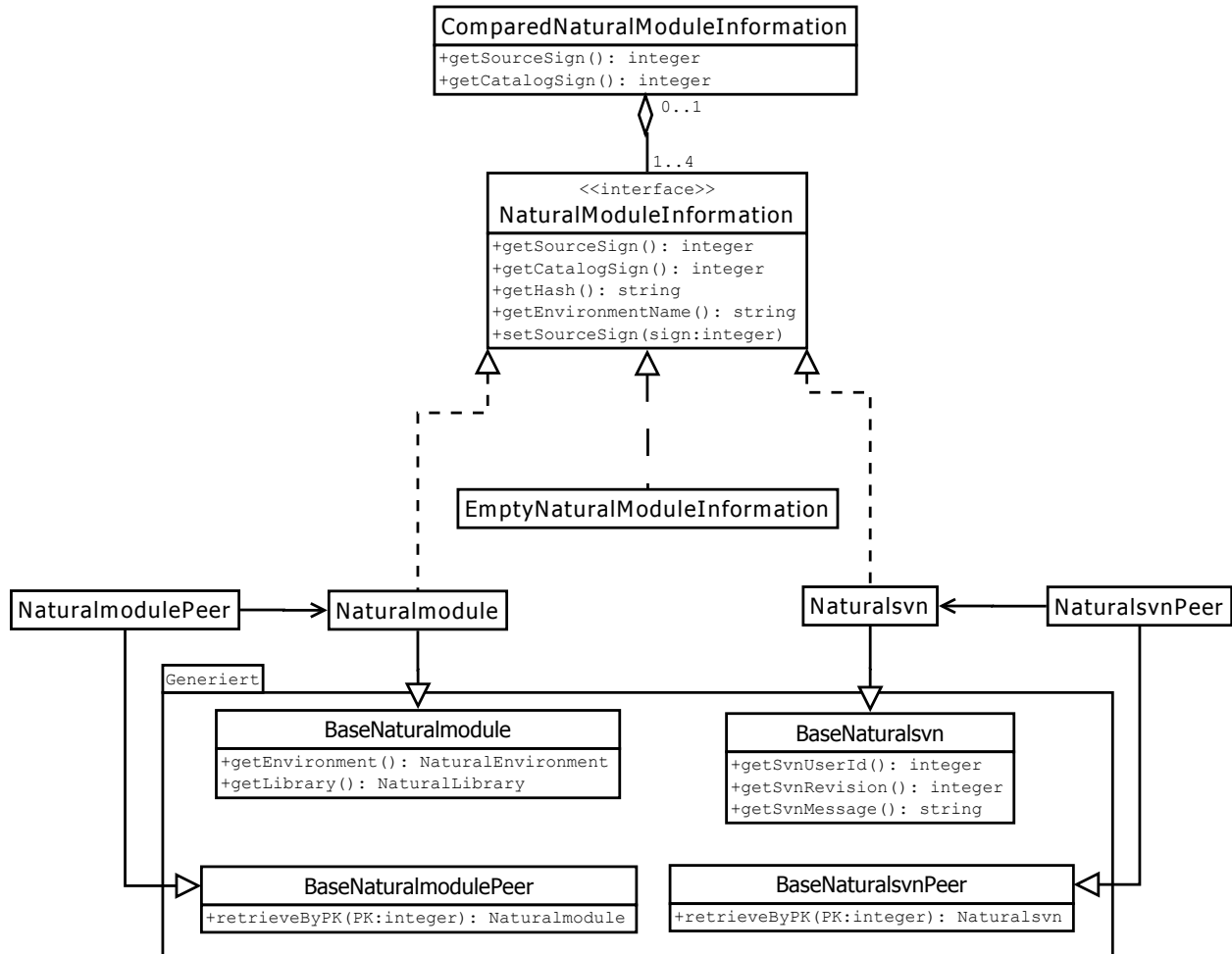







Abbildung 11: Klassendiagramm

A.12 Benutzerdokumentation

Ausschnitt aus der Benutzerdokumentation:

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