Pflichtenheft

1 Zielbestimmung
- 1.1 Musskriterien
- 1.2 Kannkriterien
- 1.3 Abgrenzungskriterien

2 Produkteinsatz
- 2.1 Anwendungsbereiche
- 2.2 Zielgruppen
- 2.3 Betriebsbedingungen

3 Produktübersicht
4 Produktfunktionen
5 Produktdaten
6 Produktleistungen:
7 Qualitätsanforderungen
8 Benutzungsoberfläche:
- *8.1 Muss-Kriterien*
- *8.2 Kann-Kriterien*

9 Nichtfunktionale Anforderungen
10 Technische Produktumgebung
- 10.1 Software
- 10.2 Hardware
- 10.3 Orgware
- 10.4 Produktschnittstellen

11 Spezielle Anforderungen an die Entwicklungsumgebung
- 11.1 Software
- 11.2 Hardware
- 11.3 Orgware
- 11.4 Entwicklungsschnittstellen

12 Gliederung in Teilprodukte
13 Ergänzungen
14 Glossar

1. Zielbestimmung

Ziel des Projektes ist die Erstellung eines Codegenerators, welcher einen funktionstüchtigen und erweiterbaren Quellcode aus einem konformen UML-Diagramm erstellt. Der Codegenerator soll auf beliebigen Betriebssystemen laufen.
Inhalt des Quellcodes soll eine browserbasierte Java-Anwendung werden, in welcher eine Lebenslaufakte von Anlagen und Geräten für erneuerbare Energien zusammengefasst wird und diese Akten beliebig bearbeitet werden können.

1.1 Musskriterien

1. Generator, welcher jedes UML-Diagramme in funktionsfähige Webanwendung mit Persistenzschicht (Datenbank) umwandelt.
2. Restriktionsbedingungen erkennen und einhalten
3. Anlegen, Löschen, Ändern und Lesen von Objekte
4. Relationen anlegen
5. Kaskadierung von Operationen
6. Navigation
7. Speichern von Dokumenten und Dateien
8. Anzeige einer Gesamtübersicht
9. Persistente Speicherung
10. Unterscheidung zwischen zwingenden und optionalen Attributen
12. Enumerationen richtig umsetzen

1.2 Kannkriterien

1. Suchfunktion
2. Editieren der Eingabemasken
3. Mehrbenutzerbetrieb

1.3 Abgrenzungskriterien

1. Unzulässige Modelle analysieren
2. Vorschlag möglicher Alternativen
3. Verweise
4. Editierungsverlauf protokollieren
5. Zugriffsschutz und Benutzerverwaltung

2. Produkteinsatz

Der Codegenerator soll einen gültigen, funktionstüchtigen Quellcode ausgeben, der auf beliebigen Rechnern funktioniert.
Firmen, die dieses System nutzen, wird ermöglicht, ihre Parks und Anlagen möglichst einfach zu verwalten, d.h. neue Anlagen zu erstellen, aktuelle Anlagen nach dem Wartungsstand abfragen, diese eventuell bearbeiten oder löschen. Zudem soll das Programm Warnungen ausgeben, falls eine Wartung nötig ist.

2.1 Anwendungsbereiche

Die Software ist so konzipiert, dass sie zu nahezu in jedem Bereich einsetzbar ist. Jedoch wurde sie für die Erstellung und Verwaltung der Lebenslaufakten von erneuerbare Energieanlagen entwickelt.

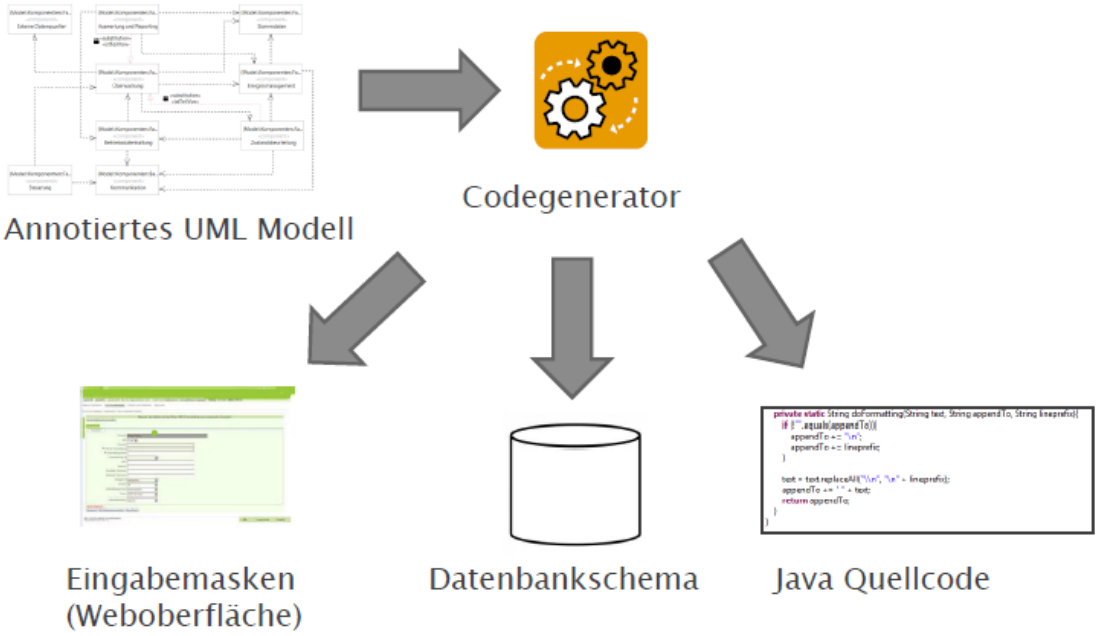
2.2 Zielgruppen

Betreiber von erneuerbaren Energieanlagen.

2.3 Betriebsbedingungen

Programme, die installiert sein müssen: JAVA mit allen Plugins, Java-Eclipse Juno.
Es muss genügend Speicher vorhanden sein und ein Webbrowser

3 Produktübersicht



4 Produktfunktionen

4.1.1 Muss-Funktionen

/LFM 10/ Codegenerator *Geschätzter Arbeitsaufwand: 30%*

Generator, welcher UML-Diagramme in funktionsfähige Webanwendung mit Persistenzschicht (Datenbank) umwandelt.

/LFM 10.10/ Annotierte UML 2.0 Modelle einladen *Relativer Arbeitsaufwand: 40%*

Die annotierten UML 2.0 Modelle sollen eingelesen werden können. Dabei sollen normgerechte Spezifikationen erkannt und entsprechend in der Ausgabe umgesetzt werden.

/LFM 10.20/ Restriktionsbedingungen erkennen *RAW: 40%*

Das Programm soll in der Lage sein, Restriktionsbedingungen, die das UML-Diagramm vorgibt (Komposition, Multiplizitäten, etc.), zu erkennen und das Anlegen von Relationen und Instanzen entsprechend zu beschränken. Integritätsbedingungen der Datenbank sind dahingehend zu erweitern. Enumerationen müssen ebenfalls eingehalten werden.

/LFM 10.30/ Kreisschluss von Restriktionsbedingungen *RAW: 20%*

Falls die Restriktionsbedingungen derart spezifiziert sind, dass aus logischer Sicht das Anlegen eines Objektes unmöglich wird, obwohl Szenarien möglich sind in denen dieses Objekt zulässig ist, so soll das Programm mögliche Alternativen bieten, wie bspw. Definition von mehreren Objekte gleichzeitig, um die Restriktionsbedingungen zu erfüllen. (Ein Bsp. für diesen Fall wäre eine 2:2 Relation zweier Klassen, sodass keine Instanz einer Klasse einzeln angelegt werden kann.)

/LFM 20/ Anlegen neuer Objekte *AW: 15%*

In der Webanwendung soll das Anlegen von Objekten gemäß der in /LFM 10/ eingegebenen Spezifikationen möglich sein. Dabei sind die Restriktionsbedingungen zu wahren.

/LFM 20.10/ Instanzen anlegen *RAW: 70%*

Insbesondere sollen Instanzen angelegt werden können. Ebenso soll es möglich sein, Attribute entsprechend der vorgegebenen Datentypen zu belegen.

/LFM 20.10.10/ Restriktionsbedingungen beim Anlegen von Instanzen

Beim Anlegen von Instanzen sind klassenspezifische Restriktionsbedingungen (Existenzabhängigkeit) entsprechend des UML-Diagramms zu beachten (u.a. betrifft dies Aggregatklassen in Kompositionsbeziehungen). Diese sollen entsprechend

überprüft werden und bei Anlegen einer Instanz in unzulässiger Weise, soll die Operation zurückgewiesen werden.

/LFM 20.10.20/ Unterscheidung zwischen zwingenden und optionalen Attributen *AW: 14%*

Wenn im UML Diagramm zwischen optionalen und zwingenden Attributen unterschieden wird, so ist /LFM 20.10/ und /LFM 30.30/ dahingehend zu beschränken, dass die gültige Belegung zwingender Attribute immer gewährleistet ist.

/LFM 20.20/ Relationen anlegen *RAW: 30%*

In der Webanwendung können Relationen entsprechend des UML-Modells zwischen Instanzen und die Attribute von Relationen definiert und angelegt werden.

/LFM 20.20.10/ Relationsrestriktionen

Entsprechend der Spezifikationen des UML-Modells soll das Anlegen von Relationen entsprechend der Restriktionsbedingungen überprüft und ggf. zurückgewiesen werden. Dabei ist insbesondere auf Multiplizitäten zu achten.

/LFM 30/ Bearbeiten vorhandener Objekte *AW: 10%*

/LFM 30.10/ Löschen von Objekten *RAW: 30%*

/LFM 30.20/ Kaskadierung von Operationen *RAW: 20%*

Bei den Bearbeitungs - Operationen sind die Integritätsbedingungen zu wahren, dazu sind Operationen ggf. durch den Datenbestand zu kaskadieren oder zurückzuweisen. In beiden Fällen sollte eine Benachrichtigung an den Nutzer erfolgen.

/LFM 30.30/ Instanzen bearbeiten *RAW: 30%*

Es soll insbesondere möglich sein innerhalb der Webanwendung bereits angelegte Instanzen zu bearbeiten, u.a. in Hinsicht auf die Attribute und Relationen. Dabei ist insbesondere auf Restriktionsbedingungen durch Multiplizitäten (beim Löschen) und auf Datentypen der Attribute (Editieren) zu achten.

/LFM 30.40/ Relationen bearbeiten *RAW: 20%*

Es soll ebenfalls möglich sein innerhalb der Webanwendung bereits angelegte Relationen zu bearbeiten, u.a. in Hinsicht auf die beteiligten Instanzen, die ggf. aktualisiert werden müssen.

/LFK 40/ Grundübersicht *AW: 10%*

Es soll in der Webanwendung möglich sein, eine Grundübersicht der angelegten Objekte zu betrachten.

/LFM 50/ Navigation *AW: 5%*

Die Navigation soll in der Webanwendung einen Zugang von allen Seiten zu allen anderen Seiten über möglichst wenig anderen Seiten gewährleisten.

/LFM 60/ Speichern von Dokumenten und Dateien *AW: 5%*

Es soll möglich seien Dateien (Dokumente, Benutzerhandbücher) Objekten (Instanzen wie Parks oder Anlage oder auch Relationen) zuzuordnen.

/LFM 70/ Objekte anzeigen *AW: 10%*

Es soll innerhalb der Webanwendung möglich sein, Objekte, welche gespeichert wurden, anzuzeigen.

/LFM 70.10/ Instanzen anzeigen *RAW: 50%*

Instanzen werden dabei insbesondere mit Ihren Attributen und Relationen dargestellt.

/LFM 70.20/ Relationen anzeigen *RAW: 50%*

Relationen werden u.a. in Hinsicht auf die beteiligten Instanzen dargestellt.

/LFM 80/ Persistente Speicherung *AW: 20%*

Alle gültigen Operationen auf Objekten (Erzeugen, Bearbeiten, Löschen) sollen persistent unter Wahrung der Integritätsbedingungen (insbesondere /LFM 20/) gespeichert werden. Ebenso ist ein Wiederaufruf der Daten zu realisieren. (siehe /LFM 80/)

4.1.2 Kann-Funktionen

/LFK 10.40/ Unzulässige Modelle analysieren *AW: 9%*
Es ist ein Verhalten des Programms im Falle der Eingabe unzulässiger UML-Diagramme zu spezifizieren, dass die Nutzung des Programms möglichst wenig einschränkt.
/LFK 30.50/ Vorschlag möglicher Alternativen *AW: 7%*
Wenn eine Operation zurückgewiesen wird, soll ein Vorschlag für gültige Operationen erfolgen.
Geschätzter Arbeitsaufwand: 7%
/LFK 40.10/ Suchfunktion *AW: 10%*
Es soll in der Webanwendung möglich sein, nach bestehenden Objekten mit bestimmten Attributwerten zu suchen.
/LFK 50.10/ Verweise *AW: 7%*
An geeigneten Stellen soll es möglich sein in der Webanwendung durch einen Verweis auf geeignete Seiten zu wechseln, so bspw. Seiten zu Instanzen, die in Relation stehen.
/LFK 70.30/ Editieren der Eingabemasken *AW: 5%*
Es soll möglich sein, den Stil der Eingabemasken anzupassen, etwa mit stylesheets (.css).
/LFK 80.10/ Mehrbenutzerbetrieb
Integritätsbedingungen sollen auch bei gleichzeitiger Nutzung der Datenbank durch mehrere Benutzer gewahrt bleiben.
/LFK 80.20/ Editierungsverlauf protokollieren *AW: 10%*
Operationen sollen protokolliert werden und dadurch zurückverfolgt werden können.
/LFK 90/ Zugriffsschutz und Benutzerverwaltung *AW: 10%*
Es soll möglich sein verschiedene Benutzerklassen zu definieren (Rollen) und die Zugriffsrechte für diese Benutzerklassen zu verwalten.

5. Produktdaten

In der UML ist festgehalten welche Daten das Programm haben wird. Allein durch das UML wird festgelegt welche Daten das Endprogramm haben wird

/LD 10/ Daten zum Objekt
Es können Daten zu jeder Klasse angelegt werden. Diese können geändert, gelöscht und ausgelesen werden.
/LD 20/ Dokumente und Dateien
Es können Dokumente und Dateien zu den Objekten hochgeladen werden.

6. Produktleistungen

Große UMLs können eine Weile dauern bis das Programm sie umgesetzt hat. Zugriffe sollen schnell durchgeführt werden können.

7. Qualitätsanforderungen

Das Hauptaugenmerk dieses Projektes beruht vor allem auf der Funktionalität und Zuverlässigkeit des Codegenerators, sowie der darin implementierten Datenbank.
Vor allem auf die Konsistenz der Datenbank ist zu beachten, da voraussichtlich im späteren Verlauf das Programm von mehreren Personen benutzt wird. Es muss verhindert werden, dass Überschneidungen, sowie nutzlose Datenreste übrig bleiben (bspw. Gelöschter Park, aber noch vorhandene Anlagen) um die Funktionalität zu gewährleisten.
Auch muss beachtet werden, dass sich die Programmstruktur aufgrund neu hinzu kommender Klassen jederzeit ändern kann.
Die Qualität der Benutzeroberfläche wird vorerst auch nur auf Funktionalität beschränkt, d.h. Es werden nur die Grundfunktionen sichtbar gemacht und eine übersichtliche Ansicht für die Akten geschaffen. Designtechnische Ansprüche werden vernachlässigt.

Produktanforderung	Sehr gut	Gut	Normal	Nicht relevant
Funktionalität	x			
Zuverlässigkeit	x			
Effizienz		x		
Benutzbarkeit			X	
Änderbarkeit		x		
Übertragbarkeit				x

8. Benutzungsoberfläche

8.1 Muss-Kriterien

/LD10/ Das Programm besitzt eine einfache Browserorientierte Oberfläche
/LD20/ Schlichtes Design mit Ansicht verfügbarer Anlagen und Parks und Suchfunktion
/LD30/ Klassen werden mit allen Details aufgelistet und eventuelle Downloadmöglichkeiten in einer Tabelle angegeben.

8.2 Kann-Kriterien

/LD40/ Der Codegenerator besitzt eine Textbox mit Archivsuche und einen Button zum Starten des Generators
/LD50/ Nach dem Start des Quellcodes wird ein Login-Formular aufgerufen

9. Nichtfunktionale Anforderungen

/NFA 10/ Erweiterbarkeit
Das Projekt soll so durchgeführt werden, dass spätere Wartung, Pflege und Erweiterungen möglichst effizient erfolgen kann. Das Qualitätssicherungskonzept spezifiziert u.a. Standards, die dies gewährleisten sollen.
/NFA 20/ Benutzbarkeit
Das Produkt soll gute Benutzbarkeit gewährleisten und übersichtlich gestaltet sein, dies wirkt sich insbesondere auf Farb-, Navigations- und Formulargestaltung aus. Die Webanwendung soll in deutscher Sprache gestaltet sein.

10 Technische Produktumgebung

10.1 Software:
Java-Eclipse Juno
Hibernate
10.2 Hardware:
Außer entsprechend großer Speicher ist keine besondere Hardware nötig
10.3 Orgware:
Auf Orgware wurde bewusst in unserem Projekt keine große Aufmerksamkeit geschenkt.
10.4 Produktschnittstellen
Webbrowser und die Eingabemaske für das UML

11 Spezielle Anforderungen an die Entwicklungsumgebung

11.1 Software
Alle Programme müssen installiert sein. Nur funktionsfähig, wenn alle installiert sind.
11.2 Hardware
-
11.3 Orgware
-
11.4 Entwicklungsschnittstellen
JAVA Eclipse Juno

12 Gliederung in Teilprodukte/Arbeitspakete

/LP1/ Erstellung des Codegenerators zur Übersetzung von UML-Diagrammen in einen funktionsfähigen Quellcode
- *Geschätzter Arbeitsaufwand: 35%*

/LP2/ Erstellung einer persistenten relationellen Datenbank
- *Geschätzter Arbeitsaufwand: 15%*

/LP3/ Tests auf Integrität der Datenbank, sowie des Codegenerators
- *Geschätzter Arbeitsaufwand: 5%*

/LP4/ Entwicklung einer Browser-basierten Benutzerschnittstelle zur Verwendung der Lebenslaufaktenverwaltung als Java-Quelltext
- *Geschätzter Arbeitsaufwand: 10%*

/LP5/ Entwicklung der Datenbank-basierenden Suchfunktion, sowie Funktionen zur Bearbeitung der Akten
- *Geschätzter Arbeitsaufwand: 5%*

/LP6/ Implementierung der Quellcodes aus /LP2/ und /LP4/ in den Codegenerator
- *Geschätzter Arbeitsaufwand: 25%*

/LP7/ Erstellung des Benutzerhandbuchs und einer ausreichenden Dokumentation
- *Geschätzter Arbeitsaufwand: 5%*

13. Ergänzungen

14. Glossar

Attribute	Eigenschaften eines Objektes, in Bezug auf Instanzen wie bspw. “Anlage” sind dies bspw. “Leistung” oder “Größe” und in Bezug auf Relationen wie “besitzt” sind dies insbesondere beteiligte Klassen aber auch die zugehörige Instanz einer Relationsklasse.
Bearbeiten	Bezeichnet das Löschen oder Editieren eines Objektes.
Beziehungen	konkrete Realisierung einer Relation.
Codegenerator	Ein Programm das bei Eingabe eines UML-Diagramms, einen Quelltext ausgibt.
Datenbank	Menge aller gespeicherten Daten.
Editieren	Bezeichnet das Bearbeiten von Attributen eines Objektes, jedoch nicht das Löschen eines Objektes.
Funktionstüchtig	Der Code heißt funktionstüchtig, wenn er ausgeführt werden kann und die beabsichtigten Funktionen implementiert sind.

Instanzen	konkrete Realisierungen einer Klasse.
Klasse	Menge von Attributen.
Löschen	Zerstört ein Objekt.
Objekte	Menge an Instanzen und Beziehungen.
Persistenzschicht	Die Persistenzschicht ist zuständig für die persistente Ablage von Objekten.
Relation	stellt eine Beziehung zwischen zwei Klassen dar.
UML-Diagramm	Ein in der Modellierungssprache UML verfasstes Klassendiagramm.
AW	geschätzter Arbeitsaufwand in Bezug auf das Gesamtprojekt
RAW	geschätzter Arbeitsaufwand in Bezug auf die übergeordnete Anforderung