Lastenheft

Inhaltsverzeichnis:

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.

2. Produkteinsatz

Der Codegenerator soll einen gültigen, funktionstüchtigen JAVA-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.

3. Produktdesign

/LD10/ Das Programm besitzt eine einfache Browserorientierte Oberfläche.

/LD20/ Der Codegenerator gewährleistet die Auswahl eines UML 2.0 - Modells inklusiver Annotationen und besitzt einen Button zum Starten des Generators.

/LD30/ Nach dem Start des Quellcodes wird ein Login-Formular aufgerufen (Zur Eingabe eines Passworts um sicherzustellen das man die Berechtigung hat auf die Webseite zuzugreifen und nicht zur Gewährleistung eines Zugriffsschutz wie beschrieben in /LFK 100/)

/LD40/ Schlichtes Design mit Ansicht verfügbarer Objekte. Die Visualisierung der Nutzeroberfläche orientiert sich am Datenmodell. Instanzen werden tabellarisch dargestellt, Beziehungen über Verweise auf andere Objekte.

/LD50/ Objekte werden mit allen Details aufgelistet und eventuelle Downloadmöglichkeiten als Links angegeben.

4. Funktionalität und Arbeitspakete

4.1 Produktfunktionen

4.1.1 Muss-Kriterien

/LFM 10/ Codegenerator *Geschätzter Arbeitsaufwand: 20%*
Es wird ein Generator ausgehende von bereits bestehenden System und Frameworks entwickelt, welcher UML-Diagramme zuerst in JAVA-Code und dann in eine funktionsfähige Webanwendung mit Persistenzschicht (Datenbank) umwandelt. Spätere Erweiterungen über die Webanwendung sind nicht möglich. Die Migration von bereits bestehenden Datenbeständen wird nicht gewährleistet.

> /LFM 10.10/ Annotierte UML 2.0 Modelle einladen *Relativer Arbeitsaufwand: 45%*
> 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: 45%*
> 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.
>
> /LFM 10.30/ Kreisschluss von Restriktionsbedingungen *RAW: 10%*
> 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 10.40/ Enumerationen korrekt umsetzen
> Enumerationen werden erkannt und es werden die Auswahlmöglichkeiten auf der Webseite zu Verfügung gestellt, die im UML angegeben wurden.

/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 20.20/ 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 20.10.20/ Unterscheidung zwischen zwingenden und optionalen Attributen
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 30/ Bearbeiten vorhandener Objekte *AW: 15%*

/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. Es soll eine Benachrichtigung an den Nutzer erfolgen, die es ermöglicht einzusehen, welche Änderungen durch diese Kaskadierung ausgeführt werden, die der Nutzer also nicht primär verlangt hat. Es soll dann dem Benutzer ermöglicht werden die Aktion abzubrechen, falls er mit der Kaskadierung nicht einverstanden ist.

/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.

/LFM 40/ Objekte anzeigen *AW: 10%*
Es soll innerhalb der Webanwendung möglich sein, Objekte, welche gespeichert wurden, anzuzeigen.

> /LFM 40.10/ Instanzen anzeigen *RAW: 50%*
> Instanzen werden dabei insbesondere mit Ihren Attributen und Relationen dargestellt.
>
> /LFM 40.20/ Relationen anzeigen *RAW: 50%*
> Relationen werden u.a. in Hinsicht auf die beteiligten Instanzen dargestellt.

/LFM 50/ Startseite *AW: 5%*
Die Startseite der Webanwendung soll übersichtlich gestaltet sein und die Möglichkeit bieten, sowohl die Seiten zum Anlegen neuer Objekte aufzurufen (siehe /LFM 20/) als auch die Seiten zur Bearbeitung bzw. Anzeige bereits bestehender Objekte aufzurufen (siehe /LFM 30/ bzw. /LFM 40/).

/LFM 60/ Navigation *AW: 10%*
Einstieg in die Webseite soll einen Überblick alles Klassen sein. Mit Klick auf eine Klasse sollen alle Instanzen der Klasse angezeigt werden. Die Navigation soll in der Webanwendung einen Zugang von allen Seiten zu allen anderen Seiten über möglichst wenig anderen Seiten gewährleisten.

/LFM 70/ Speichern von Dokumenten und Dateien *AW: 5%*
Es soll möglich seien Dateien (Dokumente, Benutzerhandbücher) bei den Objekten (Instanzen wie Parks oder Anlage oder auch Relationen), die im UML extra dafür durch Stereotypen gekennzeichnet wurden, zuzuordnen. Diese werden dann in der Datenbank gespeichert und es wird ein Link zum Download angeboten.

/LFM 80/ Persistente Speicherung *AW: 15%*
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/)

> /LFM 80.10/ Mehrbenutzerbetrieb
> Integritätsbedingungen sollen auch bei gleichzeitiger Nutzung der Datenbank durch mehrere Benutzer gewahrt bleiben.

4.1.2 Kann-Kriterien

/LFK 10.50/ 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.

/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 40.30/ Editieren der Eingabemasken *AW: 5%*
Es soll möglich sein, den Stil der Eingabemasken anzupassen, etwa mit stylesheets (.css).

/LFK 80.20/ Editierungsverlauf protokollieren *AW: 10%*
Operationen sollen protokolliert werden und dadurch zurückverfolgt werden können. Eine Art Logging-Datei wird erstellt um Änderungen nachvollziehen zu können. Ein Rollback wird nicht gewährleistet.

/LFK 100/ Zugriffsschutz und Benutzerverwaltung *AW: 10%*
Es soll möglich sein, verschiedene Benutzerklassen zu definieren (Rollen) und die Zugriffsrechte für diese Benutzerklassen zu verwalten, d.h. Den Lese- und Schreibzugriff auf für bestimmte Objekte zu gewähren / verbieten.

/LFK 110/ Suchfunktion *AW: 10%*
Es soll in der Webanwendung möglich sein, nach bestehenden Objekten mit bestimmten Attributwerten zu suchen.

4.1.3 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.

4.2 Arbeitspakete

/LP1/ Erstellung des Codegenerators zur Übersetzung von UML-Diagrammen in einen funktionsfähigen JAVA-Quellcode
- *Geschätzter Arbeitsaufwand: 20%*

/LP2/ Erstellung einer persistenten relationellen Datenbank

- *Geschätzter Arbeitsaufwand: 10%*

/LP3/ Tests auf Integrität der Datenbank, sowie des Codegenerators
- *Geschätzter Arbeitsaufwand: 10%*

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

/LP5/ Entwicklung der Schnittstelle zwischen Benutzeroberfläche und Datenbank
- *Geschätzter Arbeitsaufwand: 10%*

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

/LP7/ Erstellung des Benutzerhandbuchs und einer ausreichenden Dokumentation
- *Geschätzter Arbeitsaufwand: 15%*

5. Qualitätssicherung

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

6. 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 dauerhafte Speicherung von Objekten als Datensätze.
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