SW-Praktikum
Aufgabenbeschreibung: GBIS

1 Allgemeine Anforderungen

Die Aufgabe des Praktikums im Vertiefungsbereich Betrieblicher Informationssysteme baut auf Themen der Vorlesungen Entwicklung von Softwaresystemen I bis III auf, und soll Fähigkeiten im Umgang mit Inhalten aus dem Vertiefungsbereich verfeinern. Während der Entwicklung eines Softwaresystems werden in der Vorlesung erlernte Methoden an praktischen Aufgaben erprobt und ihre Notwendigkeit verdeutlicht. Durch vorgegebene Techniken ist der Aufwand der Bearbeitung der Aufgabe soweit eingeschränkt worden, daß die Aufgabe innerhalb des Zeitrahmens von elf Wochen zu bewältigen ist. Dennoch sollen alle Phasen eines vorgegebenen Lebenszyklus-Modells durchlaufen und gruppendynamische Aktivitäten gefördert werden.

2 Inhalt und Ziel der Aufgabe

Ziel der hier beschriebenen Aufgabe ist die Entwicklung einer Software, deren Einsatzgebiet in das Anwendungsumfeld der Leistungsbewertung Web-basierter E-Commerce-Anwendungen fällt. Zu diesem Zweck wird auf der Grundlage des TPC-W-Benchmarks ein vollständiges E-Commerce-System vorgegeben, das einen Web-basierten Buchhandel simuliert. Es sollen Messungen zur Bewertung der Leistungseigenschaften der eingesetzten Technologien ermöglicht werden. Um diese Messungen durchführen zu können, ist zuerst eine umfassende Datenbasis aufzubauen. Diese muß bestimmten Anforderungen bezüglich Verteilung, Umfang und Struktur der Daten genügen, um vergleichbare Ergebnisse zu erhalten. Die Datenbasis wird dabei durch ein Datenbanksystem verwaltet. *Das Teilsystem zur Population der Datenbank ist im Praktikum zu entwickeln.*

3 Motivation

Softwareentwicklung im Sinne des Software Engineerings bedeutet die Erstellung eines Software-Systems ausgehend von einer System-Spezifikation durch ein Team mittels vorgegebener Techniken, Methoden und Werkzeuge. Bei der Entwicklung großer, komplexer Systeme sind u. a. folgende Prinzipien für den Entwurf und die Programmierung elementar:

- *Modularisierung,*
- *Explizite Schnittstellen,*
- *Dokumentation,*
- *Grundprinzip:* ***Teile und Beherrsche***

Weiterhin sind für reale Anwendungsszenarien charakteristisch:

- *die Software muß Anforderungen Dritter genügen,*
- *die Benutzung erfolgt nicht nur durch den jeweiligen Entwickler, sondern durch eine Vielzahl weiterer Personen und*
- *die Entwicklung findet in der Regel in Teams statt, d. h., das Verhalten der Komponenten ist ausreichend zu spezifizieren und dokumentieren, um Integrations- und Koordinationsprobleme zu vermeiden.*

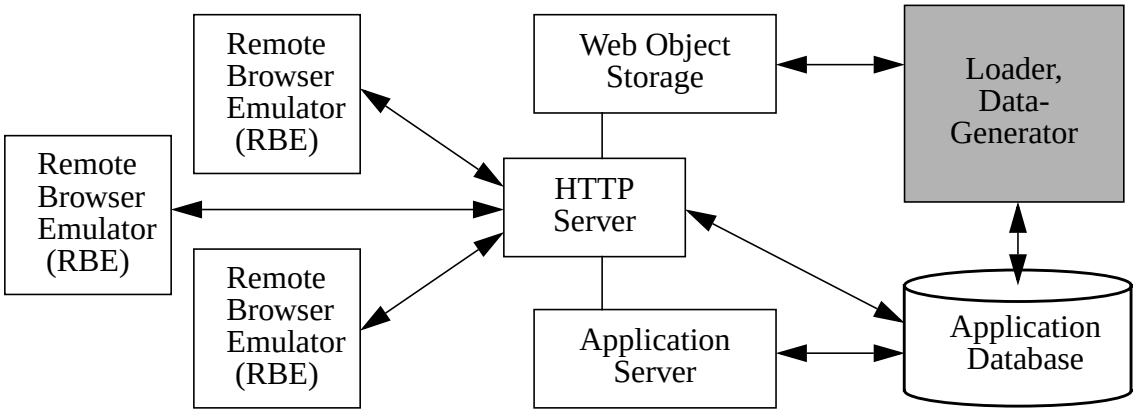
Aufgrund dieser Charakteristika wurde der TPC-W Benchmark [www.tpc.org] als zentrales Thema dieser Aufgabe ausgesucht. Seine sehr genaue und vor allen Dingen hinreichend vollständige Spezifikation der funktionalen und nicht-funktionalen Anforderungen erlaubt eine Fokussierung auf die Anwendung von Methoden zur Softwareentwicklung und die Verfeinerung von bekannten Techniken für Modellierung, Entwurf und Programmierung, ohne durch bekannte Probleme, wie z. B. Mehrdeutigkeiten in der Problembeschreibung oder Unzulänglichkeiten der Dokumentation, zu verwirren oder aufzuhalten. Das gesamte Spektrum an Freiheitsgraden während der Realisierung bzw. der Modellierung des Softwaresystems bleibt erhalten.

Durch die Beschränkung auf die Betrachtung nur eines Teilsystems und der Vorgabe des übrigen Systems entsteht die Notwendigkeit, daß sich das zu realisierende System in das Gesamtsystem einfügen muß, und es von Dritten in Form von anderen Teilsystemen oder Anwendern benutzt wird. Ein Aspekt der Realisierung betrifft somit auch die Ergonomie der Benutzerschnittstellen.

Das Teilsystem kann ferner nur durch die Leistung der gesamten Gruppe als Team realisiert werden, da der Umfang der Aufgaben kein anderes Arbeiten zuläßt. Dies verdeutlicht die Notwendigkeit der Einhaltung oben genannter Prinzipien.

4 Anwendungs-Szenario

Im Bild ist ein Überblick der Architektur gegeben. Es zeigt die notwendigen Komponenten und deren Beziehung zueinander. Von diesen Komponenten ist im Zuge des Praktikums der Loader/Data-Generator (grau unterlegt) zu entwickeln.



Eine RBE-Instanz ist ein Benutzerverhalten simulierender HTTP-Client (WWW-Browser-Simulation). Auf der Basis von vorgegebenen Wahrscheinlichkeiten werden Anfragen generiert, anschließend Antworten analysiert und daraus wieder entsprechende Anfragen erzeugt. Der RBE ist nicht nur für die Lastgenerierung, sondern auch für die Messung der Verarbeitungszeit einer Anfrage verantwortlich.

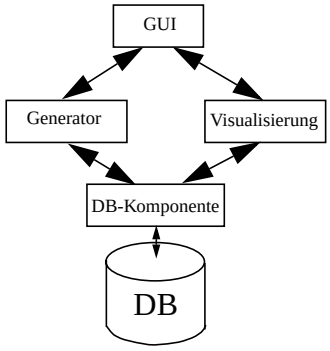
Zentrale Instanz des Systems ist der HTTP-Server, der Anfragen von RBE-Instanzen verarbeitet. Zu diesem Zweck muß er auf das Datenbanksystem zugreifen oder Verarbeitungseinheiten an den Applikationsserver delegieren. Ein Datenbankzugriff ist immer dann notwendig, wenn Bücherlisten angefordert, Suchanfragen formuliert oder Bücher geordert werden. Der Applikationsserver ist auch für die Transaktionsverarbeitung verantwortlich, die unter anderem bei verbindlichen Bestellungen notwendig wird.

Web-Objekte, wie beispielsweise Bilder, werden gesondert verwaltet. Sie können in einem Datenbanksystem verwaltet werden und müssen nicht zwingend in einem Dateisystem gehalten werden.

Das Teilsystem zur Population der Datenbank ist für die Erzeugung der Datenbasis notwendig. Zur Datenbasis gehören neben anderen Daten: Autorenverzeichnisse, Bücherkataloge, Bestellinformationen und auch Web-Objekte, wie beispielsweise Bilder. Dabei sind Vorgaben bezüglich der statistischen Verteilung der Ausprägungen einzelner Attribute zu beachten. Der TPC-W macht sehr genaue Angaben bezüglich der Häufigkeit des Vorkommens einzelner Ausprägungen.

5 Konkretisierung des Teilsystems

Das zu realisierende System besteht grob aus den nachfolgenden Komponenten, wobei in der Modellierungs- bzw. Entwurfsphase diese Aufteilung verfeinert oder durch eine geschicktere Aufteilung ersetzt werden kann. Sie dient hier nur der Verdeutlichung des Gesamtzusammenhangs. Wichtig bei der Modellierung ist die explizite Definition und Dokumentation der Schnittstellen, um eine parallele Entwicklung der entworfenen Komponenten zu unterstützen:



Die DB-Komponente stellt der Generator- und Visualisierung-Komponente eine Schnittstelle zur Verfügung, die den Datenbankzugriff kapselt und von dem Datenbanksystem abstrahiert. Es sind verschiedene Abstraktionsgrade denkbar: beispielsweise kann für jede DB-Operation (wie Select, Insert, Update, ...) eine Methode an dieser Schnittelle angeboten werden oder auch für jede Relation (Buch, Autor, Bestellung, ...) eine Methode zum Schreiben oder Lesen einer Instanz bzw. einer Menge von Instanzen definiert werden. Auch ein objektorientierter Ansatz ist vorstellbar, so daß jede Relation in einer eigenen Klasse gekapselt wird. Der Zugriff auf das Datenbanksystem selber geschieht mittels der standardisierten Anfragesprache SQL.

Die Generator-Komponente umfaßt die Funktionalität zum Erzeugen von Instanzen unter Beachtung bestimmter statistischer Annahmen in Abhängigkeit von vorgegebener Anzahl und Struktur. Die benötigten statistischen Funktionen können eventuell in einer zusätzlichen Komponente modelliert werden.

Die Visualisierung-Komponente dient zur Qualitätssicherung. Sie soll den Inhalt der Datenbank visualisieren, um die Überprüfung der Einhaltung der vorgegebenen statistischen Randbedingungen zu ermöglichen (liegen die Werte in den vorgegebenen Grenzen, sind sie gleichverteilt, usw.). Die hier benutzten Funktionen zur Überprüfung der Korrektheit von Verteilungen sollten nicht auf die Funktionen der Generator-Komponente zugreifen, um nicht falsche Ergebnisse mit gleichen, falschen Annahmen zu verifizieren.

Der Zugriff auf die letzten beiden Komponenten soll über eine grafische Benutzeroberfläche ermöglicht werden.

Da die Programmierung in Java erfolgt, wird als Schnittstelle zum Datenbanksystem JDBC eingesetzt. Als Datenbankverwaltungssystem kommt der Informix Dynamic Server 2000 zum Einsatz.

6 Übersicht über Durchführung und Ablauf

Die Durchführung der Aufgabe orientiert sich an dem Wasserfall-Modell [B. Boehm]. Dieses Lebenszyklusmodell gliedert sich in vier Phasen, welche nach dem Modell genau einmal durchlaufen werden. Es eignet sich in diesem Fall sehr gut, da die Anforderungen zu Beginn vollständig beschrieben werden können und keine Integrationsprobleme zu erwarten sind:

- *Anforderungsanalyse*
- *Systementwurf und Modellierung*
- *Kodierung*
- *Integration und Validation*

Am Ende jeder Phase ist von den Gruppen jeweils ein Dokument abzugeben, das die Ergebnisse der Phase zusammenfaßt und das weitere Vorgehen beschreibt. Im Rahmen eines Kolloquiums werden diese Ergebnisse diskutiert und bewertet.

6.1 Anforderungsanalyse

Die eigentliche Anforderungsanalyse entfällt aufgrund der Vorgabe der TPC-W-Benchmark-Spezifikation. Diese Phase dient vielmehr der Findung der eigenen Rolle eines jeden Studierenden innerhalb seiner Gruppe bzw. seines Teams und der allgemeinen Planung und Einarbeitung in die Teilaufgabe des Praktikums, so daß eine Aufgabenverteilung im Team möglich wird.

Diese Phase umfaßt höchstens eine Woche.

6.2 Systementwurf und Modellierung

Neben weiterer Vertiefung des Themas und der einzusetzenden API ist ein Entwurf zu modellieren, der den Anforderungen genügt. Die Anforderungen ergeben sich aus der Spezifikation. Architektur, Schnittstellen, Klassen, interne Repräsentation von Datenbankobjekten und weitere Strukturierung durch Einsatz von Paketen (respektive Java Pakkages) sind zu definieren und zu dokumentieren. Weiterhin sind für jede Komponente Testfälle zu entwickeln, die eine Überprüfung derselben ermöglichen.

Diese Phase umfaßt höchstens drei Wochen.

6.3 Kodierung

Die Implementierung des vorgestellten Systementwurfs verfeinert u. a. die Technik im Umgang mit Java, JDBC und SQL durch die notwendige Auseinandersetzung mit der Thematik und deren tatsächliche Anwendung.

Diese Phase umfaßt höchstens vier Wochen.

6.4 Integration und Validation

Die Komponenten sind zu einem funktionierenden System zu integrieren und zu testen. Neben dem Funktionstest zur Überprüfung der Zuverlässigkeit ist ferner das System gegen die Anforderungen zu validieren, geeignete Testfälle können direkt aus der Spezifikation abgeleitet werden. Die Überprüfung geschieht mit Hilfe der Visualisierungskomponente. Am Ende dieser Aufgabe und zugleich am Ende des SWPs steht eine Systempräsentation im Rahmen aller Gruppen eines jeden Betreuers.

Diese Phase umfaßt höchstens drei Wochen.

Eine ausführliche Beschreibung der einzelnen Teilaufgaben befindet sich in Abschnitt 9.

7 Organisatorisches

7.1 Gruppenstärke

Eine Gruppe besteht aus 6 Teilnehmern.

7.2 Pair-Programming

Die Gruppe wird weiter in drei Zweiergruppen unterteilt. Jede dieser Gruppen ist für die Umsetzung eines Teils des Gesamtsystems verantwortlich (eine sinnvolle Aufteilung ist z. B. Visualisierung/GUI, DB-Komponente, Generator). Insbesondere gelten die beiden Mitglieder der Zweiergruppe als Ansprechpartner bei Rückfragen zu ihrer Komponente. Die Zweiergruppen werden von der Gruppe zu Beginn des Praktikums selbst festgelegt und bleiben bis zum Ende bestehen. Sinn des Pair-Programming ist ein **gemeinsames** Arbeiten, kein simples Aufteilen der Aufgaben. Beide Entwickler sollen zusammen an einem Rechner arbeiten. Bei evtl. Nachfragen sollten daher auch beide Entwickler jederzeit alle Teile der ihnen übertragenen Komponente genaustens kennen. In der Programmentwicklung erfahrenere Teilnehmer sollen darauf achten, daß ihr Partner nicht auf der Strecke bleibt.
Bei Besprechungen mit den Betreuern hat jeweils *ein* Mitglied pro Zweierteam teilzunehmen; dabei sollen am Ende beide an der gleichen Anzahl Besprechungen teilgenommen haben.

7.3 Testat & Kolloquium

Nach der Phase des Entwurfs wird ein schriftliches Testat über die bisherigen Inhalte des Praktikums stattfinden. Nach der Systemintegration wird es weiterhin ein Kolloquium geben, in dem jeder Teilnehmer in einem kurzen Einzelgespräch von seinem Betreuer zum Praktikum befragt wird. Diese beiden Noten fließen in die Individualbewertung ein.

7.4 Materialien

Zu Beginn der Aufgabe werden folgende Dokumente ausgehändigt:

- *Aufgabenbeschreibung, (dieses Dokument),*
- *TPCW-Spezifikation 5.1 (Draft, Clause 1 - 6).*

Ferner stehen unter folgender WWW-Adresse weitere Informationen zum Praktikum, zu den zu verwendenden APIs und zum Informix DBMS zur Verfügung:

http://wwwdbis.informatik.uni-kl.de/courses/praktika/swp/2002/index.html

7.5 Bewertung

Die Bewertung des Praktikums setzt sich aus einer Gruppen- und einer Individualnote zusammen. Bei vollständiger Bearbeitung der gestellten Aufgaben kann eine maximale Punktzahl von 100 Punkten erreicht werden. Nach jeder Phase des Praktikums werden die Abgaben der Gruppe bewertet, die Summe der dabei erreichten Punkte ergibt die Gruppenbewertung. Diese macht insgesamt 50% der Punkte aus. Zusätzlich kann jeder Teilnehmer in den Kolloquia zusätzliche Punkte erringen, die in den Individualanteil einfließen.

Zur Bewertung der Gruppenleistung sind zu vorgegebenen Terminen Dokumente abzugeben, die die Ergebnisse der vorangegangenen Arbeit dokumentieren und erläutern. Ferner beinhalten sie Antworten zu Fragen, bzw. Aufgaben, die zusätzlich zur Hauptaufgabe der Realisierung eines Software-Systems vor oder während der Praktikumsaufgabe gestellt werden.

Alle anzufertigenden Dokumente sollen nach HTML transformiert werden können, so daß sie auf einem Web-Server bekannt gemacht werden können. Welche Werkzeuge zum Erstellen der Dokumente eingesetzt werden, ist freigestellt. Es gilt lediglich die Anforderung, daß das Werkzeug die Erzeugung einer HTML-Repräsentation unterstützen sollte. Ferner ist eine druckbare Version der Dokumentation der Aufgabenbearbeitung bereitzustellen (pdf/ps).

Die absolute Mindestanforderung für einen erfolgreichen Abschluß des Praktikums des Praktikums besteht darin, zum Ende ein lauffähiges System vorzustellen, das zumindest eine Relation des vorgegebenen DB-Schemas mit sinnvollen Daten füllt. **Als Programmiersprache ist ausschließlich JAVA (JDK 1.x, JDBC 1.x) zu verwenden!**

7.6 Systemumgebung

Die Systemumgebung wird unter der oben genannten URL beschrieben. Es ist für jeden ein Benutzeraccount eingerichtet, die wiederum zu Gruppen zusammengefaßt werden. Für jede Gruppe wird ein Verantwortlicher bestimmt, auf dessen Account ein eigenes lokales WWW-Verzeichnis für Dokumentationen angelegt wird. Die Einstiegsseite wird dann über einen Link auf der zentralen Seite zugänglich gemacht. Die Seiten können lokal von allen gelesen werden, sind aber nicht im gesamten WWW verfügbar. Die Gruppe kann die Verfügbarkeit bei Bedarf an ihre eigenen Vorstellungen anpassen, die Seiten sollten aber auf alle Fälle für die Betreuer und die anderen Gruppen lesbar sein.

8 Schlußbemerkung

Das Praktikum wurde konzipiert, um die in den ersten Semestern vorgestellten Techniken und Methoden praktisch anzuwenden. Aufgrund der beschränkten Zeit von 11 Wochen kann natürlich nur ein Teil davon berücksichtigt werden. Nichtsdestotrotz ist die Aufgabe sehr gut geeignet, um ein Gefühl für die Probleme bei der Entwicklung eines großen Systems im Team zu entwickeln.
Die benutzten Werkzeuge und die Programmierung in JAVA sollten (hoffentlich) schon aus den Übungen zu den Vorlesungen bekannt sein, so daß sich der Einarbeitungsaufwand hierfür in Grenzen hält.

Und nun: Viel Spaß beim Entwickeln :-)

9 Aufgaben

9.1 Hauptaufgabe

Die Hauptaufgabe besteht in der Realisierung eines Software-Systems zur Population der Datenbank laut TPCW-Spezifikation. Konkret sind die Paragraphen 1 und 4 der Spezifikation zu erfüllen. Paragraph 1 beschreibt die Anforderungen an das Datenbankschema und die Datenbank-relevanten Aspekte, während Paragraph 4 die Anforderungen an die zu generierenden Daten definiert. Eine Datenbank mit entsprechendem Schema ist vorgegeben und muß nicht modelliert werden.

Die für das zu realisierende Software-System relevanten funktionalen Anforderungen finden sich in Paragraph 4 und müssen erfüllt werden.

Alle Anforderungen in Paragraph 4, die sich nicht direkt auf das zu realisierende Software-System beziehen, beispielsweise Anforderungen an den Web-Server etc., können ignoriert werden. Ferner ist es nicht erforderlich den Punkt 4.6.2.12 zu erfüllen. Somit müssen zunächst keine Bilder (Images) oder Verkleinerungen (Thumbnails) erzeugt werden. **Grafische Objekte brauchen nicht unterstützt zu werden!**

9.2 Aufgaben in den Entwicklungsphasen

Das vorgegebene Lebenszyklusmodell gliedert sich in vier Phasen, welche nach dem Modell genau einmal durchlaufen werden. Die Durchführung der Aufgabe orientiert sich an diesen Phasen:

- *Anforderungsanalyse*
- *Systementwurf und Modellierung*
- *Kodierung*
- *Integration und Validation*

In jeder Phase ist die Bearbeitung zu dokumentieren.

9.2.1 Anforderungsanalyse

Die eigentliche Anforderungsanalyse entfällt aufgrund der Vorgabe der TPCW-Benchmark-Spezifikation. Vielmehr ist sich hier in die Spezifikation und Dokumentation zu vertiefen und eine Strategie zu entwickeln, um eine Datenbasis schnell, effizient und mit korrekten Daten aufzubauen.

Zeit:

Diese Phase umfaßt eine Woche.

Zweck:

Verteilung der Aufgaben innerhalb der Gruppe,
Einarbeitung in die Thematik,
Planung des allgemeinen Vorgehens und der grafischen Benutzerschnittstelle.

Aufgaben dieser Phase:

Arbeiten sie die Spezifikation durch und entwickeln sie eine Strategie, die beschreibt, wie die benötigten Daten zu erzeugen sind. Erläutern sie kurz, wie im Allgemeinen bei der Population der Datenbank vorzugehen ist und wie im Speziellen Bestellungen behandelt werden müssen. Berücksichtigen sie dabei die Abhängigkeiten, die sich durch Fremdschlüsselbeziehungen oder andere Beziehungen aus dem DB-Schema ergeben, und die eventuell lange Laufzeit eines Generierungsdurchgangs. Überlegen sie, welche funktionalen Anforderungen die grafische Benutzerschnittstelle konkret erfüllen muß, und entwerfen sie entsprechende Interface-Flow Diagrams.
Entwickeln sie jeweils eine Formel für die Abschätzung des Speicheraufwandes und für die Anzahl der zu erzeugenden Objekte in Abhängigkeit von der Anzahl der emulierten Browser (EB) und Produkte (Items).

Schätzen sie den Aufwand für
EB = 1, Items = 1 000 und
EB = 10, Items = 100 000 ab.

Dokumente:

System-Anforderungs-Beschreibung, die oben genannte Strategie, Interface-Flow Diagrams und Abschätzungen umfaßt und dokumentiert, wer für welche Aufgabengebiete, bzw. Komponenten verantwortlich ist.

9.2.2 Systementwurf und Modellierung

Im weiteren Verlauf ist ein System zu entwerfen, das den Anforderungen der Spezifikation genügt. Dabei ist den Prinzipien eines objektorientierten Entwurfs zu folgen.

Zeit:

Diese Phase umfaßt drei Wochen.

Zweck:

Erstellung eines objektorientierten System-Entwurfs
Erarbeiten von Testfällen

Aufgaben dieser Phase:

Architektur, Schnittstellen, Klassen, interne Repräsentation von Datenbankobjekten und weitere Strukturierung durch Einsatz von Paketen (respektive Java Packages) sind zu definieren und zu dokumentieren. Entwickeln sie für jede Komponente Testfälle. Beschreiben sie das dabei zu erwartetende Verhalten. Bei der GUI ist z. B. ein Testfall „QUIT Button wird gedrückt“ denkbar. Das erwartete Verhalten ist das Schließen der Anwendung (ohne Blue Screen oder Core-Dump).

Dokumente:

System-Entwurfs-Beschreibung (mit Klassendiagramm nach UML)
Testplan (für jede Komponente 10 Testfälle mit Verhaltensbeschreibung)

Bitte besonders die Entwurfshinweise in Abschnitt 10 auf Seite 13 beachten!

9.2.3 Kodierung

In der Realisierungsphase ist der Entwurf zu implementieren. Für die Programmierung ist die objektorientierte Programmiersprache JAVA zu verwenden.

Zeit:

Diese Phase umfaßt höchstens vier Wochen.

Zweck:

Kodierung

Aufgaben dieser Phase:

Realisieren sie den von ihnen vorgestellten Entwurf unter besonderer Berücksichtigung der Einhaltung der Kardinalitäten der zu erzeugenden Objektmengen. Beschreiben sie, wie die Kardinalität der Menge von Bestellpositionen eingehalten wird. Benutzen sie für die Datenbank-Anbindung die JDBC-Schnittstelle. Für Testläufe sollte die Konfiguration (EB = 1, Items = 1000) genutzt werden.

Benutzen sie für die Kode-Dokumentation das JDK-Tool „javadoc“!

Testen sie weiterhin ihre Komponenten mit den von ihnen beschriebenen Testfällen. Verhält sich alles wie erwartet?

Dokumente:

kommentierter Kode,
Kode-Beschreibung (HTML, mit javadoc erzeugt),
Protokoll Komponententest

9.2.4 Integration und Validation

Die Komponenten sind zu einem funktionierenden System zu integrieren und zu testen. Neben dem Funktionstest zur Überprüfung der Zuverlässigkeit ist ferner das System gegen die Anforderungen zu validieren. Geeignete Testfälle können aus der Spezifikation direkt abgeleitet werden. Dies geschieht mit Hilfe der Visualisierungskomponente.

Die Realisierung ist am Ende der Praktikumsaufgabe im Rahmen einer Systemvorstellung zu präsentieren.

Zeit:

Diese Phase umfaßt höchstens drei Wochen.

Zweck:

Integration,

Test, Validation,

Ausführung: Population der Datenbank. EB=10, Item=100000

Aufgaben dieser Phase:

Fügen sie alle Komponenten zu einem vollständigen System zusammen. Testen sie es zunächst mit der Konfiguration (EB = 1, Item = 1 000) und dann mit der Konfiguration (EB = 10, Item = 100 000). Messen sie die Zeit, die zum Füllen der Datenbank nötig ist. Validieren sie das System gegen die Anforderungen, indem sie geeignete Anfragen an das DBMS stellen und diese visualisieren.

Testen sie ihr System mit geeigneten Testfällen. Arbeiten alle Komponenten fehlerfrei zusammen? Denken sie über Transaktionskontrolle und Recovery nach! Was geschieht bei einem vorzeitigen Abbruch der Verarbeitung? Sind sie gezwungen, von vorne zu beginnen, oder können sie einzelne Schritte rückgängig machen, um in einem konsistenten Zustand die Verarbeitung fortzusetzen.

Welche Möglichkeiten bieten DBVS, um die Anforderungen an die spezifischen Eigenschaften der Attribute und ihrer Ausprägungen zu gewährleisten?

Dokumente:

Abschluß-Dokumentation (Integrations- und Validations-Protokoll mit Testfällen und Ergebnis),

Dokumentation des Einsatzes der Visualisierungs-Komponente

System-Vorführung

10 Entwurfshinweise

10.1 Allgemein

Überlegen sie, ob nicht die Verwendung von Design Patterns sinnvoll ist:

- ***Erzeugungsmuster***
  - *Abstrakte Fabrik (Abtract Factory)*
  - *Erbauer (Builder)*
  - *Singleton*
- ***Strukturierungsmuster***
  - *Adapter*
  - *Fliegengewicht (Flyweight)*
- ***Verhaltensmuster***
  - *Beobachter (Observer)*
  - *Besucher (Visitor)*
  - *Strategie (Strategy)*

Anstelle von globalen Variablen kann man beispielsweise Singletons einsetzen. Es müssen nicht alle hier aufgeführten Entwurfsmuster verwendet werden. Es können auch weitere Muster eingesetzt werden.

10.2 DB-Zugriffskomponente und Generator-Komponente

Versuchen sie folgender Intention zu folgen:

- *Trennung zwischen der Strategie zur Erzeugung und dem Vorgang der eigentlichen Erzeugung, sowie der tatsächlichen Repräsentation von Objekten,*
- *Abstraktion von konkreten Tabellen des DB-Schemas und der Anfragesprache SQL,*
- *Kapselung der Datenbankschnittstelle, um später auch andere DBS unterstützen zu können.*

10.3 Grafische Benutzeroberfläche

Die grafische Benutzerschnittstelle sollte nicht nur die Möglichkeit geben, die Verarbeitung zu starten, zu beenden und zu unterbrechen, sondern auch Informationen über deren Fortschritt zur Verfügung stellen. Dazu ist es notwendig etwas über den Zustand anderer Klassen, bzw. ihrer Objekte zu erfahren. Beachten sie aber die Kapselung der Objekte! Prüfen sie daher auch die Einsatzmöglichkeit des Observer/Observable-Interfaces von Java.

10.4 Visualisierungs-Komponente

Es ist eine Komponente zu erstellen, die den Inhalt der erzeugten Datenbank visualisiert. Sie kann eingesetzt werden, um die Datenbank gegen die Anforderungen aus der Spezifikation zu validieren. Hierbei ist zweierlei zu beachten.

Zum einen soll die Verteilung der Attribute überprüft werden. Es sollte daher möglich sein, statistische Information über das Minimum, das Maximum, den Mittelwert, Anzahl der von Null verschiedenen Tupel und Anzahl der voneinander verschiedenen Werte zu erhalten. Anhand einer entsprechenden grafischen Darstellung soll ferner gezeigt werden, daß die gewählte Zufallsfunktion wirklich eine Gleichverteilung der Attributwerte erzeugt, d. h., jede Ausprägung ist mit ungefähr der gleichen Anzahl vorhanden. Zur Darstellung sind geeignete Diagramme anzubieten (z. B. Torten-, Linien-, Balken- oder Blockdiagramme) und dem Benutzer zur Auswahl anzubieten. Da es gilt, eine sehr große Anzahl von Daten auszuwerten, sind sinnvolle Maßnahmen zur Skalierung vorzusehen. Auch die Auswahl eines Anzeigebereichs („Zoom-Funktion“) erscheint hier sinnvoll. Stichproben, also die Auswahl eines zufälligen Tupels aus einer vorgegeben Tabelle, sollten ebenso unterstützt werden, wie die gezielte Auswahl anhand einer ID. Die grafischen Elemente und Realisierungsaspekte sind dabei allerdings der grafischen Benutzeroberfläche zuzuordnen.

10.5 Metadaten-Verwaltung

An verschiedenen Stellen des Systems ist es notwendig, auf Informationen über das zugrundeliegende Schema zuzugreifen. Beispielsweise soll in der GUI bei der Visualisierung der Daten bei Auswahl einer Tabelle automatisch eine Auswahlbox mit den zugehörigen Attributen gefüllt werden. Es hat sich hier als äußerst praktisch erwiesen, diese Metadaten in einer eigenen Hilfskomponente bereitzuhalten, die bei Bedarf diese Informationen liefern kann. Da es sich dabei um Schemainformationen handelt, wird diese Komponente sinnvollerweise der DB-Komponente angegliedert. Die Implementierung kann in einer recht primitiven Weise erfolgen, jedoch sollte dabei darauf geachtet werden, daß Änderungen im Schema nachgezogen werden müssen.

11 Dokumentationsrichtlinien

11.1 Grundprinzipien

Prinzip 1: Verständliche Darstellung

- *angemessene Notation und Strukturierung*
- *Vollständigkeit / Konsistenz*

Prinzip 2: Minimale Darstellung

- *Nicht alle möglichen, sondern nur notwendige Punkte beschreiben.*

Prinzip 3: Modulare Struktur

- *Komplexität eines Software-Systems sollte sich gleichmäßig auf verschiedene Ebenen des Entwurf, bzw. auf verschiedene Komponenten oder Module verteilen.*
- *„Teile und Beherrsche!“*
- *Kopplung von Komponenten sollte minimal sein.*
- *Komponenten-Kohäsion (interne Kopplung) sollte maximal sein.*
- *Information-Hiding / Kapselung sollte maximal sein.*

Prinzip 4: Vertikale Verfolgbarkeit

- *Verschiedene Dokumente unterschiedlichen Abstraktionsgrades müssen zueinander konsistent sein.*
- *Beziehungen zwischen Anforderungen, Entwurf und Realisierung müssen klar und explizit dokumentiert werden.*

Prinzip 5: Horizontale Verfolgbarkeit

- *Verschiedene Sichten innerhalb einer Abstraktionsebene müssen nachvollziehbar und konsistent sein. (Beispielsweise muß im Klassendiagramm und in der Beschreibung des Verhaltens Namenskonformität herrschen.)*

Prinzip 6: Dokumentation der Verifikation

- *Dokumentation der Verifikation des Entwurfs gegen die Anforderungen*
- *Dokumentation der Verifikation des Programmkodes gegen den Entwurf*

Prinzip 7: Dokumentation der Validation

- *Dokumentation der Validation des ausführbaren Systems gegen die Anforderungsbeschreibung (Spezifikation)*

11.2 Dokumentation des Entwurfs

- *Es wird ein objektorientierter Entwurf erwartet.*
- *Es ist ein Klassendiagramm zu erstellen und eine Darstellung nach UML zu wählen.*
- *Es sind die Schnittstellen der Klassen zu definieren.*
- *Es ist die Funktion einer Klasse und das Verhalten ihrer Methoden kurz zu erläutern.*
- *Die Abhängigkeiten zwischen einzelnen Komponenten bzw. Klassen sind zu dokumentieren.*

- *Nach den Prinzipien 1 und 2 sollte die System-Entwurfs-Beschreibung kurz, aber prägnant sein und alles wichtige enthalten. Details können im Kode dokumentiert werden.*

11.3 Dokumentation des Kodes

- *Es ist der Quelltext in einer für das Verständnis hinreichenden Art und Weise zu dokumentieren.*
- *Es ist jede Klasse, jede Methode und jedes Attribut in Funktion, Bedeutung und Verhalten zu dokumentieren.*
- *Der Quelltext ist so mit Anmerkungen zu versehen, daß mit Hilfe des JDK-Tools „javadoc“ automatisch eine Dokumentation in HTML erstellt werden kann. Diese Dokumentation ist im lokalen WWW-Verzeichnis dem Betreuer zugänglich zu machen.*
- *Die Dokumentation beschränkt sich nicht nur auf die Signatur, sondern bezieht sich auch auf Anweisungen innerhalb von Methoden. Es sollen wichtige Stellen im Quelltext erläutert werden (beispielsweise Fallunterscheidungen, Schleifen o. ä.), um die Lesbarkeit zu erhöhen.*