CNAM Software-Technik WS 2012/2013
Praktikum

Joachim Hauck
Alexander Figul
lehre@melsicon.de

26. November 2012

I Das Projekt „Bibliotheks-Verwaltung“

Ziel des Projektes ist es, der Bibliothek der Hochschule Darmstadt ein neues System zu bauen, welches sie in der Verwaltung ihres Bestandes und der Ausleihvorgänge durch Studenten unterstützt. Dabei gelten folgende Bedingungen:

- Als Hochschul-Einrichtung verfügt die Bibliothek nicht nur über einen großen Bestand an Büchern, sondern auch über eine Vielzahl von anderen Medien: wissenschaftliche Zeitschriften, Videos oder auch CDs.
- Da die Fachbereiche der Hochschule über Darmstadt und Dieburg verteilt sind und jeweils über ganz spezifische Literaturbedürfnisse verfügen, besitzt die Bibliothek verschiedene Sammlungen an den unterschiedlichen Standorten. Dadurch können natürlich Redundanzen entstehen: Ein Lehrbuch über die numerische Mathematik ist sowohl für die Mathematiker, als auch für die Informatiker relevant.
- Die Bibliothek hält all ihre Werke vor, damit die Angehörigen der Hochschule, sprich also in erster Linie die Studenten und Dozenten, damit arbeiten können. Eine Grundvorrausetzung dafür ist eine funktionierende Suchmöglichkeit, die den genauen Standort eines Exemplars mitteilen kann.
- Der Bestand verändert sich laufend und muss entsprechend aktualisiert werden:

  - neue Medien zu aktuellen Themen oder in neuren Auflagen werden dazu gekauft
  - alte, überholte Medien werden aussortiert
  - ausgeliehene Medien, die weit über den vorgegebenen Zeitraum nicht mehr zurückgegeben wurden, werden abgeschrieben
- Ausschließlich Hochschulzugehörige dürfen die Bibliothek benutzen. Dabei existiert auch hier eine große Fluktuation:
  - Jedes Semester kommen neue Studenten an die Hochschule, während alte exmatrikuliert werden und sie verlassen.
  - Angestellte besitzen zwar typischerweise eine längere Zugehörigkeit, doch auch hier kommen immer wieder neue hinzu, während bestehende die Hochschule verlassen.
- Damit mit einem Medium gearbeitet werden kann, ist es häufig nicht ausreichend, ausschließlich während der Bibliotheks-Öffnungszeiten und in deren Räumen darauf zugreifen zu können. Darum existiert die Möglichkeit, Medien für eine gewisse Zeit auszuleihen. Hierüber muss genauestens Buch geführt werden, da ohne Erinnerung die Ausleihfristen häufig überschritten würden.
- Ein verliehenes Medium steht in der gesamten Verleihzeit nur dem Ausleiher zur Verfügung, während ein Medium, welches sich in der Bibliothekt direkt befindet, in der gleichen Zeit von vielen benutzt werden kann. Damit wichtige, häufig nachgefragte Werke jederzeit verfügbar sind, werden manche Exemplare als „Bestandsexemplare“ markiert und dürfen nicht verliehen werden.

II Erwarteter Umfang der Implementierung

Die Implementierung des Projektes soll zumindest die funktionierende Kernfunktionalität enthalten. Sprich: Medien sollen erfasst, verändert, gesucht und gefunden sowie ausgeliehen und zurückgegeben werden.
Unerheblich ist dabei die Form der Ein- bzw. Ausgabe des Programmes. Ein Kommandozeilen-Programm mit simplem Menü ist ebenso hinreichend wie eine graphische Oberfläche oder ein Programm, welches Dateien einliest und ausgibt.

Jegliche über die Kernfunktionalität hinausgehenden Features, wie zum Beispiel eine besonders detaillierte Suche oder nützliche Übersichten über die überfälligen, aber noch nicht zurückgegebenen Medien, ist nicht Teil des erwarteten Umfanges der Implementierung. Dies bedeutet jedoch ausdrücklich nicht, dass Sie sich in Ihren Gedanken und Entwürfen auf den notwendigen Part beschränken sollen. Selbst, wenn einige Elemente nicht fertig realisiert werden sollten, so können sie doch wertvolle Anregungen darstellen, die das Design des Systems positiv beeinflussen.

Diejenigen Teile, die von Ihnen implementiert werden, sollen mit ihren Entwürfen konsistent sein, soweit dies in Hinsicht auf einen eventuell reduzierten Funktionalitätsumfang gegeben sein kann.

III Erwartete Abgabeartefakte

Das Praktikum wird am Ende des Semesters benotet. Neben dem Eindruck Ihrer Arbeitsweise, den Sie während der Termine hinterlassen, stellt vor allem Ihre Projekt-Mappe das zentrale Bewertungskriterium dar. In ihr sollen alle Arbeitseegebnisse - bis auf die Implementierung - festgehalten werfen. Um eine Nachvollziehbarkeit Ihrer Vorgehensweise zu gewährleisten, passen Sie auch die Ergebnisse der vergangenen Praktika an, sofern dies neue Erkenntnisse nötig machen!

Insgesamt erwartet werden folgende Artefakte:

Ihre Projektmappe In Ihrer Projektmappe sollen alle Arbeitsergenisse Ihrer Analyse- und Design-Phase enthalten sein. Damit umfasst sie am Ende des Praktikums Ihre

- Use Cases als Use-Case-Diagramm sowie die detailierte Ablauf-Beschreibung der von Ihnen exemplarisch ausgewählten Use Cases.
- Das Analyse-Klassendiagramm.
- Die dynamischen Diagramme, also zumindest jeweils ein
  - Sequenzdiagramm
  - Aktivitätsdiagramm
  - Zustandsdiagramm
- Das Design-Klassendiagramm.

Das Ziel der Projektmappe ist hierbei die Möglichkeit, Ihre Gedankengänge zumindest annähernd nachvollziehen zu können. Dies kann nie vollständig erreicht werden, da die jeweiligen Arbeitsergebnisse immer nur einen kleinen Teil ihrer Entscheidungen und deren Grundlagen abzubilden vermögen. Trotz alle dem wird Konsistenz von Ihrer Arbeit erwartet. Mit anderen Worten: Das Design ihres Systems soll dem Bild entsprechen, was Sie in ihrer Analyse gezeichnet haben.

Das bedeutet auch: Wenn Sie im Verlauf des Praktikums bemerken, dass ein schon fertig gestellter Teil nicht mehr Ihren Ansprüchen entspricht, so passen Sie ihn an. Die Projektmappe bildet am Ende das Gesamt-Ergebnis ab. Wichtig sind demnach also die jeweils letzten Versionen der Arbeitsergebnisse, nicht vergangene Zwischenstände.

Ihr lauffähiges Programm in einer Form, die eine Ausführung leicht möglich macht. Wenn Sie auf Fremd-Software, z.B. eine Datenbank, zurückgreifen, stellen Sie bitte auch eine passende Installations-Datei sowie -Anleitung bereit.

Ihr Quellcode der dem lauffähigen Programm zu Grunde liegt.

Ihr MagicDraw-Projekt um eine vollständige Nachvollziehbarkeit Ihrer Arbeitsweise zu gewährleisten.

IV Aufbau des Praktikums

Im Verlauf des CNAM-SWT-Praktikums soll ein kompletter Projekt-Durchlauf stimuliert werden, so dass die in der Vorlesung besprochenen Methoden erprobt werden können.

Dabei kann grob in folgende Phasen unterteilt werden:

- 1. Termin: Festlegung des Projekt-Gegenstandes, Analyse
- 2. Termin: Design
- 3. Termin: Test und Implementierung
- 4. Termin: Review und Abgabe

Die Implementierungsphase, die an anderer Stelle oft im Vordergrund steht, tut es hier ausdrücklich nicht! Viel mehr soll besonderer Wert auf die Analyse- und Design-Vorgänge gelegt werden.

1 Aufgabenstellung für den ersten Termin

1. Machen Sie sich Gedanken darüber, wie ein System, welches den zuvor festgelegten Projekt-Gegenstand realisiert, Ihrer Meinung nach sein müsste. Legen Sie dabei primär Wert auf die Belange der späteren Anwender, nicht auf die technische Umsetzung.

Halten Sie Ihre Überlegungen in Form eines UML-Use-Case-Diagramms fest. Gestalten Sie dabei zumindest einige ihrer Use-Cases detailiert, in dem Sie auch einen entsprechenden Workflow festhalten.

2. Entwerfen Sie auf dieser Basis ein UML-Analyse-Klassendiagramm. Falls Sie sich dabei von Ihren ursprünglichen Ideen entfernen, passen Sie auch Ihre Use-Cases entsprechend an!

Fertigen Sie zuerst einen Entwurf auf Papier an. So kann sicher gestellt werden, dass kein Interpretationsmonopol entsteht. Denn einen Computer kann immer nur eine Person bedienen. Damit liegt, oft unbewusst, die Entscheidung, welche genaue Ausprägung eine Idee annimmt, nur in einer Hand innerhalb ihres Zweierteams. Dies wird vermieden, indem auf das Medium Papier zurückgegriffen wird. Mit Hilfe von zwei Stiften ist hier ein echtes, paralleles Verändern möglich.

Sobald Sie sich grob auf eine Version des Klassendiagramms geeinigt haben, realisieren Sie es auch mit MagicDraw. Unterscheiden Sie dabei möglichst nach

- Boundary-Klassen: Schnittstellen zum Anwender oder zu anderen Systemen.
- Controller-Klassen: die Geschäftslogik, der „handelnde Teil“ des Programmes.
- Entity-Klassen: Klassen, die lediglich zur Datenerhaltung dienen.

2 Aufgabenstellung für den zweiten Termin

1. Dynamische Diagramme:

Wählen Sie Ihre drei wichtigsten Use-Cases (diejenigen, die die zentralen Aspekte des Systems aufzeigen) und erstellen dafür

- ein Sequenzdiagramm
- ein Zustandsdiagramm
- ein Aktivitätsdiagramm

Ein Diagramm soll dabei jeweils ein Use Case detailiert beschreiben. Natürlich sollte für jedes Diagramm ein Use Case gewählt werden, welches damit auch sinnvoll beschrieben werden kann.

Ein Sequenzdiagramm beschreibt den gesamten Lebenszyklus eines Objektes – von der Erzeugung, über die Verwendung und Interaktion mit anderen Objekten bis hin zur Zerstörung des Objektes. Zustands- wie auch Aktivitätsdiagramm erfassen das Verhalten von Objekten. Dabei beschreibt das Zustandsdiagramm ein Objekt, welches ereignisgetrieben – also von Aktivitäten, die außerhalb seines direkten Einflußbereiches liegen – gesteuert wird. Ein Aktivitätsdiagramm beschreibt hingegen ein Objekt, dessen Verhalten vom Ergebnis der eigenen Aktivitäten abhängt.

Erstellen Sie auch hier wieder – analog zum Entwurf des Analyse-Klassendiagramms im letzten Termin – zuerst einen händischen Entwurf auf Papier!

2. Design-Klassendiagramm:

Überführen Sie Ihr Analyse-Klassendiagramm in ein Design-Klassendiagramm. Berücksichtigen Sie dabei Ihre Überlegungen bezüglich der von Ihnen erstellen dynamischen Diagramme.

In diesem Diagramm müssen nicht nur alle Attribute und Methoden spezifiziert werden, ihre Notation muss auch bezüglich der verwendeten Datentypen auf die gewählte Implementierungstechnologie angepasst werden.

Erzeugen Sie neben den für die Geschäftslogik notwendigen Methoden auch:

- Getter-/Setter-Methoden sowie
- Konstruktoren

Damit eine Klasse zu jedem Zeitpunkt die alleinige Kontrolle über alle ihre Attribute behalten kann, werden Getter- und Setter-Methoden verwendet. Dabei gestatten Getter-Methoden anderen Objekten einen lesenden, Setter-Methoden einen schreibenden Zugriff auf die Eigenschaften. Wird sowohl eine Getter, wie auch eine Setter-Methode für ein Attribut bereitgestellt, können die gleichen Manipulationen von außen vorgenommen werden wie sie bei einer Eigenschaft

mit der Sichtbarkeit „public“ möglich wären. Anders als bei einer solchen besitzt die Klasse allerdings die Möglichkeit, alle Aktionen zum einen zu registrieren und zum anderen bei Bedarf auch abzuwandeln. Dadurch, dass jede Änderung letztlich durch die Klasse selbst vorgenommen wird, besitzen die Eigenschaften einen genau definierbaren Zustand, während bei einem public-Attribut nie exakt vorhergesagt werden kann, welche Form der Inhalt zwischenzeitlich angenommen hat.

Konstruktoren hingegen sind spezielle Methoden, die aufgerufen werden, sobald eine neue Klasseninstanz erzeugt werden soll. Ihnen können verschiedenste Parameter übergeben werden, anhand derer typischerweise die Eigenschafts-Inhalte des neues Objektes bestimmt werden. Ihre Aufgabe ist also die Initialisierung eines neuen Objektes.

Von zentraler Bedeutung sind weiterhin die Beziehungen (Assoziationen und Aggregationen) zwischen den Klassen inklusive ihrer Muliplizitäten. Beziehungen bilden die Grundlage für das Zusammenspiel der einzelnen Programmteile. Denn innerhalb einer Klassen-Methode kann nur auf die Objekte zugegriffen werden, die der Klasse auch bekannt sind. Zwar können prinzipiell auch beliebige Objekte als Parameter eines Methoden-Aufrufes übergeben werden, wodurch eine kurzzeitige, lose Kopplung zwischen den beteiligten Objekten hergestellt wird.

Als sinnvoller erscheinen allerdings an vielen Stellen festere Verbindungen. Eine Klasse, die ein Bank-Konto abbilden soll, wird in der Verwendung höchstwahrscheinlich exakt einem Besitzer zugeordnet. Da ein Konto eher selten den Besitzer wechselt, kann im Normalfall diese Kopplung schon bei Erzeugung des Konto-Objektes hergestellt und über den gesamten Lebenszyklus unverändert beibehalten werden. Die verarbeitende Geschäftslogik kann nun, sobald sie über ein Konto-Objekt verfügt, immer auch ohne Umwege auf das dazu passende Besitzer-Objekt zugreifen, ohne, dass ihr dieses extra zugeführt werden müsste. Gleiches gilt auch für den umgekehrten Weg: Eine Person kann typischerweise beliebig viele Konten bei einer Bank besitzen. Eine entsprechende Klasse wird demnach über eine Liste mit den jeweiligen Konto-Objekten verfügen.

Allgemein formuliert: Besitzt ein Beziehungsende eine Multiplizität von 1, bedeutet dies, dass immer nur ein einzelnes Exemplar der verbundenen Art mit einer Instanz in Beziehung stehen kann. Dementsprechend wird in der Klasse mit diesem Beziehungsende ein Attribut benötigt, dass eine Referenz auf die andere beteiligte Klasse realisiert. Soll ein Beziehungsende mit einer höheren Multiplizität als 1 umgesetzt werden, muss statt einer einzelnen Referenz eine Liste aus Referenzen verwendet werden. Die Erzeugung der jeweiligen Attribute innerhalb der Klassen übernimmt MagicDraw beim Anlegen bzw. Verändern einer Beziehung.

Besondere Formen der Assoziation sind Aggregation und Komposition. Während eine Assoziation nur aussagt, dass sich die Instanzen zweier Klassen „kennen“, also in einer beliebigen Art und Weise miteinander in Beziehung stehen, spezifiziert die Aggregation diese Verbindung genauer. Sie gibt an, dass eine Menge von Objekten eine bestimmte Art Objekt bildet. Als Beispiel kann ein Motor

dienen: Er besteht aus einer Vielzahl von Einzelteilen, die auch alle für sich selbst als selbstständiges Teil mit jeweils spezifischen Eigenschaften und Funktionen betrachtet werden können. Richtig zusammengesetzt ergeben sie allerdings eine neue Einheit, ein Ganzes, das auch mehr sein kann als nur die Summe seiner Teile.

Eine Komposition widerum ist eine spezielle Form der Aggregation. Sie wird verwendet, wenn ein Objekt nur in Zusammenhang mit der übergeordneten Einheit überhaupt einen Sinn und damit eine Daseinsberechtigung erwirbt. Während die Schrauben eines Motors auch ohne ihn in anderem Zusammenhang Verwendung finden können, gestaltet sich dies bei einer einer Bemerkung zu einem Bestellvorgang in einem Online-Shop anders. Zwar kann ihr eine gewisse Eigenständigkeit zugesprochen werden: Eine Bestellung kann schließlich mehrere verschiedene Bemerkungen besitzen, die sich voneinander nicht nur in Erstellungsdatum, Inhalt und Autor unterscheiden. Trotzdem aber besteht eine starke Abhängigkeit zur Bestellung. Verschwindet diese z.B. weil sie aus dem System gelöscht wurde, gibt es keinen Grund, weiterhin die dazugehörigen Bemerkungen zu speichern.

3 Aufgabenstellung für den dritten Termin

1. Generieren Sie mit MagicDraw aus Ihren Modellen die passenden Quellcode-Vorlagen.

2. Realisieren Sie ausgewählte Aspekte des Systems in Java oder C++, indem Sie die generierten Klassenrümpfe vervollständigen.

Welcher Funktionsumfang genau als Mindestmaß betrachtet wird, um ein Testat zu erzielen, entnehmen Sie bitte der Projektbeschreibung (siehe: *II Erwarteter Umfang der Implementierung*).

Gehen Sie bei der Implementierung nach der Test-First-Methode vor. Das bedeutet: Erstellen Sie immer zuerst einen Unit-Test, der die neue Funktionalität prüft, bevor Sie sie realisieren!

Der Vorteil dieser Methode ist, dass Sie sich vor der Realisierung Gedanken darüber machen müssen, was die Methode am Ende im einzelnen tun soll und wie sie das Ergebniss ihrer Arbeit bestätigen können. Sie laufen so nicht Gefahr zu viel zu implementieren und die Qualität ihres Codes steigt. Der Test dient außerdem als Dokumentation über die Absicht Ihrer Methode. Anhand eines Tests könnte beispielsweise Ihr Partner genau so viel implementieren, wie zum Bestehen des Tests erforderlich ist. Tauschen Sie hier ruhig öfter die Rollen.

Sinnvoll ist zudem in kleinen Schritten vorzugehen. Der Entwicklungszyklus sieht dann so aus: Ein kleiner Test der fehlschlägt $\Rightarrow$ Soviel Code das er läuft $\Rightarrow$ wieder eine kleiner Test der fehlschlägt $\Rightarrow$ wieder soviel Code das er läuft.

Sie werden merken, dass Sie durch die Test gezwungen sind sich mehr mit Implementierungsdetails auseinanderzusetzen und dadurch Ihre Fähigkeiten in ihrer Programmiersprache schnell verbessern werden.

Wie Überall ist es auch in unserem Projekt keine gute Idee auf Grund von Zeitdruck auf Tests zu verzichten!

4 Aufgabenstellung für den vierten Termin

Der letzte Praktikumstermin dient der Durchführung der Reviews als ‚simulierte Abnahme'. Dabei reviewen sich jeweils zwei Praktikumsgruppen gegenseitig und fertigen ein Reviewprotokoll an, das von allen Beteiligten zu unterschreiben ist. Das unterschriebene Reviewprotokoll ist der Praktikumsmappe beizufügen.

Zum Review gehört auch die exemplarische Demonstration des lauffähigen Programms auf den Workstations des Labors oder auf Ihrem Laptop. Im Reviewprotokoll ist die Durchführung dieser Prüfung zu vermerken.

Bei ‚abnahmeverhindernden' Mängeln ist eine Beseitigung noch vor der Abgabe der Arbeitsartefakte durchzuführen. Sofern keine ‚abnahmeverhindernden' Fehler festgestellt werden, geben Sie am Ende der Reviewsitzung Ihre Projektmappe mit allen Arbeitsergebnissen bei den Betreuern ab. Unterschreiben Sie gemeinsam mit den Reviewern zuvor das Reviewprotokoll.

Falls ‚abnahmeverhindernde' Fehler festgestellt werden (z.B. kein lauffähiges Programm), sind diese bis zum 08.02.2013 zu beheben und an lehre@melsicon.de zu senden. Bei einer späteren Abgabe ist das Praktikum nicht bestanden.