Aufgabenblatt 4

Ziel dieses Blatts

Nun wird es langsam Ernst mit der konkreten Arbeit an Design und Implementierung des eigenen Projekts. Nach den intensiven Vorarbeiten, die im Rahmen der ersten drei Aufgabenblätter zu absolvieren waren, sind Sie (hoffentlich) gut vertraut mit der Problematik Ihres Themas. In diesem Arbeitsblatt werden

- einerseits die Rahmen für die organisatorische Planung der Implementierungsphase abgesteckt
- und andererseits die Anforderungen an einen ersten Designentwurf Ihres Modells spezifiziert.

Definitions- und Entwurfsphase (Designentwurf)

Dieses Thema nimmt den meisten Raum im [Balzert] ein, so dass Sie die einschlägigen Konzepte kennen. Da die zu verwendenden Architekturkonzepte weitgehend klar bzw. in der Themenstellung vorgegeben sind, liegt der Schwerpunkt dieser Etappe auf der Definitionsphase und dort genauer auf der OOA [Balzert, LE 13].

Aufgabe:

Entwickeln Sie stufenweise und in mehreren Iterationsrunden nach der im [Balzert] vorgegebenen Methodik das Modell Ihres Systems und implementieren Sie dieses in Rational Rose.

Entwerfen Sie zunächst als **statisches Modell** Ihres Systems die Klassen, Attribute, Assoziationen und Vererbungsstrukturen sowie die Zuordnung der Klassen zu Paketen und halten Sie die Gründe für Ihre Entscheidungen als Teil der zu erstellenden Design-Beschreibung fest (so viel oder wenig detailliert, dass ein Neueinsteiger diese Entscheidungen nachvollziehen kann ohne sich in Details zu verlieren). Erstellen Sie dazu geeignete UML-Diagramme.

Entwickeln Sie auf dieser Basis das **dynamische Modell** Ihres Systems. Spielen Sie an Hand geeigneter Anwendungsfälle und Szenarios alle wichtigen Interaktionskonstellationen durch, identifizieren Sie die Methoden, welche die einzelnen Klassen zur Verfügung stellen müssen und überlegen Sie, wie sich der Fluss der Botschaften durch das System gestaltet. Erstellen Sie dazu geeignete UML-Diagramme und ergänzen Sie die Design-Beschreibung entsprechend.

Beachten Sie, dass die beiden Modellierungsphasen im Sinne eines evolutionären, iterativen Entwicklungsprozesses mehrfach durchlaufen werden (siehe auch die Ausführungen zur Implementierung) und dass den Phasen in den beiden Themen unterschiedliches Gewicht zukommt[1]. Aktualisieren Sie den Designentwurf im Rhythmus der Wochenscheiben (siehe unten).

Abzugeben: Eine erste Version der Design-Beschreibung (im ps- bzw. pdf-Format, Umfang max. 8 Seiten, 11pt Schriftgröße), aus welcher alle wichtigen Aspekte sowohl des statischen als auch des dynamischen Modells der ersten Iteration Ihres Systems ersichtlich sind. Halten Sie sich dazu an die im Blatt 2 angegebene Gliederung einer Design-Beschreibung mit dem Schwerpunkt auf den Abschnitten 3 und 4. Nutzen Sie sowohl textuelle als auch grafische Elemente (UML-Diagramme geeigneten Typs), um die Logik Ihres Systems darzustellen.

[1] „Die entstehende Spezifikation besteht aus einem statischen und einem dynamischen Modell. Welches dieser beiden Modelle in der Systemanalyse das größere Gewicht besitzt, hängt wesentlich von der jeweiligen Anwendung ab. Das statische Modell ist bei Datenbankanwendungen besonders wichtig. Das dynamische Modell ist bei stark interaktiven Systemen sowie bei technischen Systemen von besonderer Bedeutung.“ [Balzert, S. 378]

Implementierung

Designentwurf, Implementierung und Tests werden in der Praxis nicht nach dem „Wasserfallmodell“ durchlaufen, sondern sind – mit leichter zeitlicher Versetzung – ineinander verzahnt. Das erleichtert es, Designentscheidungen schon in einer frühen Phase auf praktische Realisierbarkeit hin zu überprüfen und sie gegebenenfalls in einem weiteren Iterationsschritt mit nicht zu hohem Aufwand zu korrigieren.

Implementierung, Dokumentation und Tests sollen deshalb im Praktikum zeitnah den Iterationen des Designentwurfs folgen, um die Vollständigkeit und praktische Realisierbarkeit der entwickelten Konzepte zu sichern. Es ist besonders wichtig, dabei arbeitsteilig vorzugehen, um die gesetzten zeitlichen Rahmen einzuhalten und mit den (personellen) Gruppenressourcen sparsam umzugehen. Dazu muss die Arbeit genau geplant und sinnvoll aufgeteilt werden. Extreme Programming[2] (XP) schlägt dazu verschiedene Arbeitstechniken vor, die im Praktikum Anwendung finden sollen:

- die Aufteilung des Projekts in „Stories“ (Prinzip der „Small Releases“),
- das „Pair-Programming“ (http://www.pairprogramming.com) und
- das Prinzip „Tests first“.

Der Aufteilung der Implementierung in **Stories** liegt der Gedanke zu Grunde, nicht erst alles zu programmieren und am Ende festzustellen, dass nichts zusammen funktioniert, sondern das System als eine Folge von Release-Versionen zu entwickeln, deren Funktionalität in Wochenscheiben Schritt für Schritt erweitert wird. Zum Abschluss der Wochenscheibe wird die geleistete Arbeit zu einem neuen lauffähigen (!) Release mit erweiterter Funktionalität integriert. Das hat den Vorteil, dass bereits während der Entwicklung dem Kunden der Prototyp mit eingeschränkter Funktionalität vorgeführt werden kann. In der Praxis wirkt sich das vertrauensfördernd auf das Verhältnis zum Kunden aus und erlaubt es, dessen Wünsche und Vorstellungen in die evolutionäre Designentwicklung mit einzubeziehen. Zugleich kann mit der Auswahl und Abgrenzung der Stories berücksichtigt werden, welche Projektteile mit welcher Priorität zu behandeln sind.

Der Ansatz des **Pair-Programming** beruht auf der Beobachtung, dass ein Zweierteam leistungsfähiger ist und qualitativ besseren Code erstellt als zwei einzelne Programmierer in der gleichen Zeit. Es gibt für diese auf den ersten Blick verblüffende Beobachtung viele Begründungen[3]. Im Praktikum soll das praktisch erprobt werden, indem Sie Ihre Gruppe für die verschiedenen Wochenscheiben in verschieden zusammengesetzte Zweierteams aufteilen, die wohl abgegrenzte Programmieraufgaben übertragen bekommen.

Aufgabe:

Eine **Story** ist also ein Implementierungsauftrag, den eines Ihrer Zweierteams (PP) in einer Woche realisieren kann. Die PP’s sind neben dem zu leistenden Beitrag zum Quellcode des Systems auch für die Zuarbeiten verantwortlich, die zur Ergänzung des Designentwurfs, der Dokumentation sowie der Suite der Komponententests erforderlich sind. Zum Ende jeder Wochenscheibe wird Bilanz gezogen, Code, Designentwurf, Dokumentation und Testsuite zu einem neuen Release integriert und dem Tutor vorgeführt.

Den erforderlichen Vorlauf des Designentwurfs eingerechnet stehen Ihrer Gruppe für die Implementierungsarbeiten vier Wochenscheiben (9.6. – 6.7.) zur Verfügung. Erstellen Sie einen (vorläufigen) Plan, welche PP’s (in wöchentlich wechselnder Zusammensetzung!) mit der Ausführung welcher Stories beauftragt werden.

Abzugeben: Planung der Stories und Untersetzung mit PP’s.

[2] siehe http://www.xprogramming.com
[3] Sie können ja selbst einmal die kreativitätsfördernde Situation der „pair programming simulation“ ausprobieren, auf die http://www.pairprogramming.com verweist.

Testkonzept

Als ein Maß für die Qualität der ausgeführten Implementierung hat es sich bewährt, die einzelnen Komponenten sowie das Gesamtsystem ausführlichen Tests zu unterziehen. Diese beiden Aspekte, *Komponententests* und *Systemtests*, sind bereits vor und während der verschiedenen Iterationen der Systemgenese vorzubereiten, um alle „Ecken" des Systems auch wirklich zu erwischen.

Für die **Komponententests** stehen mit dem JUnit-Framework (siehe Aufgabenblatt 2) Konzepte und Werkzeuge zur Verfügung, mit denen derartige Tests weitgehend automatisiert durchgeführt werden können. Es ist wichtig, die auszuführenden Testreihen genau zu planen, um die Gewähr zu haben, dass bei späteren Modifikationen am Design bzw. Quelltext alle explizit und implizit bestehenden Abhängigkeiten und Voraussetzungen auch weiterhin berücksichtigt sind.
Das XP-Paradigma „Tests first" verlangt deshalb, dass zu jeder Story bereits vor Beginn der Implementierungsarbeiten ein Satz von Testbeispielen angelegt wird, die vom zu erstellenden Code ohne Beanstandungen abgearbeitet werden. Letzteres ist zugleich ein wichtiger Qualitätsparameter für den „Erfolg der Story". Der Satz von Testbeispielen wird dann in die Testsuite integriert, gegen die alle folgenden Release-Versionen des Systems stabil laufen müssen.
Literatur: http://junit.sourceforge.net

Aufgabe:

Komponenten- und Systemtests sind fester Bestandteil des Integrationsprozesses der fertigen Stories in das Gesamtsystem. Deren Planung, Vorbereitung und Ausführung koordiniert die/der **Verantwortliche für Tests** Ihrer Projektgruppe. Dazu ist ein **Testkonzept** zu erstellen, das entsprechende konzeptionelle und organisatorische Festlegungen fixiert. Erstellen Sie innerhalb Ihres Projekts eine eigene Java-Klasse `tests`, in der Sie alle Testbeispiele aufsammeln, damit sich diese einfach vom restlichen Quelltest trennen lassen.

Abzugeben: Das Testkonzept (max. 2 Seiten)

Dokumentationskonzept

Die Dokumentation des Designs Ihres Systems besteht aus folgenden Komponenten:

- der Design-Beschreibung,
- den mit `javadoc` extrahierbaren Informationen,
- weiteren Blockkommentaren und der Inline-Kommentierung des Quelltexts
- sowie dem Quelltext selbst.

Für die Auslieferung kommen noch, soweit erforderlich, eine Anwender- sowie eine Installationsdokumentation hinzu.

Die **Design-Beschreibung** soll der aus Aufgabenblatt 2 bekannten Gliederung folgen und eine Einführung in das Design geben, die es außenstehenden Personen mit entsprechenden Java-Kenntnissen (Wartungspersonal, neuen Team-Mitgliedern) erlaubt, sich schnell mit allen wichtigen Aspekten Ihres Softwaremodells vertraut zu machen. Dazu soll die Design-Beschreibung ausführliche Informationen zur Paketstruktur und wesentliche Informationen zu den statischen und dynamischen Aspekten des Modells enthalten, die mit geeigneten grafischen UML-Diagrammen untersetzt sind.

Aufgabe:

Die Aktualisierung von Design und Dokumentation soll zeitnah zum Designentwurf und den Implementierungsarbeiten erfolgen („Prinzip der integrierten Dokumentation"). Dazu ist es erforderlich, entsprechende Zuarbeiten von allen PP's einzufordern und diese in Gruppenmeetings zu diskutieren sowie dafür die technischen und organisatorischen Voraussetzungen zu schaffen.
Für die Projektteile, die unter CVS-Kontrolle stehen, ist letzteres kein Problem. Die Integration der Zuarbeiten für das Rose-Modell (V. Design) sowie die Design-Beschreibung (V. Dokumentation)

bedürfen jedoch genauerer Überlegungen und Absprachen, die im Dokumentationskonzept festzuhalten sind.

Abzugeben: Das Dokumentationskonzept (max. 2 Seiten)

Aufwandserfassung

Bisher spielten Aspekte des Projektmanagements im Praktikum eine untergeordnete Rolle und konzentrierten sich auf technische Fragen wie die Verwendung von CVS. Für die qualifizierte Führung eines Projekts spielt jedoch auch die Aufwandsschätzung, -analyse und -erfassung eine wichtige Rolle, um die personellen Projektressourcen effizient und den jeweiligen Fähigkeiten angemessen einzusetzen.
Für die weitere Arbeit soll deshalb ein **Aufwandserfassungssystem** eingeführt werden. In der Methodik folgen wir dabei [Balzert „Software-Qualitätsmanagement“, LE 8]:

- Die auszuführenden Arbeiten werden in Teilaufgaben zerlegt und zugeordnet (V: Projektleiter).
- Jedes Teammitglied führt selbst Buch über die aufgewendete Zeit (ein Strich pro Stunde) und bewertet den Aufwand (A) sowie die Schwierigkeit (S) der Teilaufgabe auf einer Skala 1 .. 5 (1 = viel zu hoch, 2 = zu hoch, 3 = angemessen, 4 = zu niedrig, 5 = viel zu niedrig). Dazu gehört natürlich auch die Zeit für Projekttreffen und ähnliche Abstimmungs- und Diskussionsrunden.
- Der Projektleiter sammelt diese Informationen wöchentlich ein, prüft sie im Team auf Plausibilität und fasst sie in einer (fortzuschreibenden) ASCII-Datei `Aufwand.txt` zusammen, aus der ersichtlich wird, welche Teammitglieder für welche Teilaufgaben wie viel Zeit verwendet haben und wie die Arbeit eingeschätzt wurde.
- Eine Kopie der Datei wird zu den Enden der Wochenscheiben über das Submit-Verzeichnis abgegeben.

Aus dieser Aufstellung soll sowohl der Aufwand als auch das arbeitsteilige Vorgehen beim Lösen der einzelnen Aufgaben deutlich werden.
Beginnen Sie die Aufstellung mit einer Übersicht, wie Sie bisher arbeitsteilig vorgegangen sind, d.h. welche Gruppenmitglieder welche der bisherigen Aufgaben wesentlich inhaltlich bearbeitet haben.

Abzugeben: Eine Kopie der aktuellen Version der Datei `Aufwand.txt`.

Abgabe der Materialien zu diesem Arbeitsblatt

bis zum 9.6. wie üblich durch Hinterlegen aller geforderten Materialien im Verzeichnis `Submit` Ihres Gruppenaccounts auf `pcai003.informatik.uni-leipzig.de`. Beachten Sie, dass die eingereichten Materialien nach dem Ablauf der Bearbeitungsfrist aus diesem Verzeichnis **verschoben** werden.