Aufgabenblatt 3

Ziel des Blattes

Nachdem die Rollen verteilt sind und Sie sich nach der Recherche mit den grundlegenden Anforderungen Ihres Projekts vertraut gemacht haben kann es nun eigentlich losgehen. Um ein größeres Projekt erfolgreich und termintreu zu bewältigen ist es sinnvoll, noch einige Energie auf die korrekte Aufstellung des Teams zu verwenden, so dass neben der eigentlichen Software-Entwicklung auch die begleitenden Maßnahmen – insbesondere das Qualitätsmanagement und die Dokumentation – im Blickfeld bleiben.

Neben dem **Abschluss der Projektspezifikation** in Lastenheft und Glossar, die Gegenstand des ersten Reviews sein werden, sollen Sie sich deshalb in Ihrem Team über die **Grundanforderungen an das Projektmanagement** verständigen und diese in entsprechenden Dokumenten verbindlich fixieren. Als erste Fingerübung ist weiterhin eine kleine **Softwarestudie** zu erstellen, in der Sie die zu verwendenden Technologien schon einmal exemplarisch einsetzen. Dazu gibt es ein Zusatzblatt, in dem dieser Teil der Aufgabenstellung themenspezifisch konkretisiert ist.

Softwarequalität und Dokumentation

Die **Qualität einer Software** wird neben ihren funktionalen Leistungsparametern auch wesentlich durch die Qualität des Codes bestimmt. Ein Maß für diese Qualität ergibt sich aus den verschiedenen Geschäftsanforderungen, die in unterschiedlichen Phasen des Softwarelebenszyklus entstehen. Dabei stehen Fragen des Bugfixings, der Wartbarkeit und Änderbarkeit im Vordergrund, für welche es wichtig ist, dass sich auch andere, projektfremde Entwickler schnell in Designaspekte der Software einarbeiten können.

Die Einhaltung einer Reihe von **allgemeinen Prinzipien** unterstützt diese Anforderungen:

- der Quellcode ist gut strukturiert, lesbar und folgt den "Regeln für guten Code",
- Quellcode und Design sind gut dokumentiert, sowohl intern als auch extern,
- zu den einzelnen Softwareeinheiten existiert Testmaterial, das alle wesentlichen Funktionen und Zweige abdeckt.

Es macht Sinn, sich an bereits aufgestellten Coding Conventions zu halten (z.B. für Java zu finden unter http://www.oracle.com/technetwork/java/index-135089.html oder http://www.torsten-horn.de/techdocs/java-codingconventions.htm).

Es ist sowohl ein Aspekt der Selbstkontrolle als auch Aufgabe des Softwarequalitätsmanagements, die Einhaltung dieser Regeln zu überwachen. In Ihrem Team sind Verantwortliche für Dokumentation und Test festgelegt. Deren Aufgabe besteht darin, die erforderlichen Qualitätsstandards festzulegen und deren Einhaltung zu organisieren und zu kontrollieren. Dazu ist ein Dokumentationskonzept und (in einem späteren Arbeitsblatt) ein Testkonzept zu vereinbaren, schriftlich zu fixieren und später zu einem **Qualitätssicherungsplan** weiter zu entwickeln.

Die **Dokumentation** des Designs und des Quellcodes Ihres Projekts soll sich aus zwei wesentlichen Komponenten zusammensetzen: 1. der internen, quelltextnahen Dokumentation, die z.B. bei Java mit **`Javadoc`** (Siehe http://www.oracle.com/technetwork/java/javase/documentation/writingdoccomments-137785.html) extrahiert werden kann, sowie einer **speziellen Design-Dokumentation**, in der alle wichtigen Designaspekte verständlich dargelegt und begründet sind.

Ein zu erstellendes **Testkonzept** soll für die Komponententests einzelner Softwareeinheiten die **Werkzeuge eines Testframeworks** (bei Java wäre das JUnit, siehe http://junit.org/) einsetzen.

Ziel von Komponententestfällen ist es, einen Indikator dafür zu haben, ob das System bzw. einzelne Komponenten so funktionieren wie sie sollen. Dazu dient eine Vielzahl von Testbeispielen, die alle mehr oder weniger wichtigen funktionalen Aspekte und Programmzweige abdecken sollen und (in der Regel) *vor* dem eigentlichen Programmieren der Applikation oder der Änderungen erstellt werden. Die Beschreibung der Testfälle ist Bestandteil des zu erstellenden Testkonzepts.

Diese Testfälle müssen nach jeder Änderung am System durchgearbeitet (d.h. getestet) werden.

Es wird überdies stark dazu geraten, dass das Team Tests automatisieren soll. Hierzu bietet sich z.B. Jenkins (http://jenkins-ci.org/) an. Jenkins ist ein Werkzeug zur kontinuierlichen Integration.

Für die Kommunikation und Qualitätssicherung im Team macht es ebenso Sinn, Werkzeuge wie z.B. Redmine (http://www.redmine.org/) einzusetzen. Falls Sie kein großes Ticket-System benötigen, greifen Sie auf den Issue Tracker von Bitbucket oder Gitlab zurück. Eine weitere Alternative wäre das Projekttool Asana (https://asana.com/).

Die Installation dieser Werkzeuge an sich bringt allerdings sehr wenig. Vielmehr sind *Absprachen* über den Gebrauch im Team zu treffen. Diese sind zu dokumentieren und selbstverständlich regelmäßig auf Einhaltung zu prüfen!

Aufgaben:

1. Machen Sie sich mit den Regeln für guten Java-Code (und HTML) vertraut und befolgen Sie diese in Ihrem Projekt.
2. Machen Sie sich mit dem Dokumentationskonzept Ihrer Programmiersprache vertraut.
3. Machen Sie sich mit dem Unit-Framework für Komponententests vertraut.
4. Überlegen Sie den Einsatz von Werkzeugen zur Unterstützung der Softwareerstellungs- und Qualitätssicherungsprozessen und Dokumentieren Sie ihre Entscheidungen bzw. Festlegungen.

Abzugeben: Dokumentationskonzept, in dem Qualitätsstandards, Vorgehensweisen und Verantwortlichkeiten für die verschiedenen Teile der Dokumentation fixiert sind.

Es gibt keine weitergehenden allgemeingültigen Ansätze, um Softwarequalität unabhängig von Einsatzgebiet und Aufgabenstellung zu sichern. Allerdings existieren für einzelne Anwendungsbereiche Erfahrungen, welche Designprinzipien und Architekturkonzepte im jeweiligen Gebiet besonders erfolgreich waren und deshalb weit verbreitet sind. Neue Projekte ziehen doppelten Nutzen, wenn sie solchen Konzepten folgen: die Konzepte bürgen einerseits bereits für eine gewisse Qualität der Produkte und sind andererseits auch den Entwicklern vertraut, die später mit dem Code zu tun haben werden. Zu denken ist im Rahmen dieses Projektes an die Benutzung von irgendwelchen **Design Patterns**.
Einarbeitung in das Fachkonzept

Zur Einarbeitung in das Fachkonzept ist eine kleine Softwarestudie anzufertigen, in der einige grundlegende Aspekte des Themas schon einmal prototypisch realisiert werden sollen. Die Aufgabenstellung ist themenspezifisch in einem Zusatzblatt weiter untersetzt.
Gehen Sie bei der Erstellung der Softwarestudie arbeitsteilig vor:

- Starten Sie ein neues Projekt, das Sie auch gleich entsprechend versionieren (Einsatz von Git).
- Überlegen Sie sich gemeinsam, wie das Design des Prototyps aussehen soll und verteilen Sie die Implementierungsarbeiten.
- Wenden Sie das vereinbarte Dokumentationskonzept bereits in dieser Aufgabenstellung an.
- Erstellen Sie eine Designbeschreibung des Prototyps. *(Testmaterial sowie Testautomation oder sogar continous integration wird für diese Studie nicht gefordert.)*

Die **Designbeschreibung** enthält alle wichtigen Informationen zu den Struktur- und Entwurfsprinzipien Ihrer Software, welche ein durchschnittlicher, bisher mit dem Projekt nicht befasster Programmierer, kennen sollte, bevor er sich in Ihre Software einarbeitet um Änderungen oder Ergänzungen des Quellcodes vorzunehmen (und dabei die detailliertere, etwa Javadoc basierte Dokumentation des Quellcodes zu nutzen).

In der Designbeschreibung sind also alle wichtigen Struktur- und Entwurfsentscheidungen zu begründen. Erläutern Sie dabei insbesondere, mit welchen Struktur- und Entwurfsentscheidungen Sie welche Konzepte und Ansätze umgesetzt haben.
Gliedern Sie die Beschreibung in folgende Hauptpunkte:

1. Allgemeines
2. Produktübersicht
   - Beschreibung der äußerlichen Funktionsmerkmale Ihrer Anwendung
3. Grundsätzliche Struktur- und Entwurfsentscheidungen für Ihre Anwendung
   - was sollte ein bisher unbeteiligter SW-Entwickler über das Gesamtsystem gelesen haben, ehe er sich Gliederungspunkt 4. und der javadoc-Dokumentation zuwendet? Insbes.: Wie sieht die Gesamtarchitektur der Anwendung aus?
4. Grundsätzliche Struktur- und Entwurfsentscheidungen der einzelnen Pakete/Komponente der Anwendung
   - was sollte er gelesen haben, nachdem er Gliederungspunkt 3. kennt und ehe er sich der javadoc-Dokumentation eines speziellen Pakets zuwendet?

Aufgabe:

Machen Sie die obige Designbeschreibung und implementieren Sie Ihr Design, d.h. codieren Sie die Anwendung.

Abzugeben (bzw. für die Betreuer zugänglich zu machen) sind:

- Die Designbeschreibung der Softwarestudie
- Code/HTML-Dateien entsprechend der Aufgabenstellung
- Präsentation Ihrer Lösung auf den Webseiten der Gruppe. Gemeint ist hier eine irgendwie funktionierende Demo

Abschluss der Anforderungsanalyse

Aufgabe:

Auf der Basis Ihrer Recherche (Aufgabenblatt 2) sind die Rahmen für Ihre eigene Projektarbeit abzustecken und in Lastenheft, Glossar und (später) Pflichtenheft zu fixieren. Halten Sie sich dabei eng an die Gliederungsforderungen in [1]. Das Lastenheft soll mehr Funktionalität beschreiben als im Rahmen des Projekts implementiert werden kann. Gliedern Sie dazu die Muss- und Kann-Ziele sowie den Teil "Produktfunktionen" des Lastenhefts weiter auf in mehrere Etappen, die evtl. von Nachfolgeprojekten in Angriff genommen werden können.

Abzugeben: Glossar und Lastenheft

Review des Lastenhefts und der Designstudie

Zu Glossar und Lastenheft wird das **erste Review** stattfinden, in welchem der Umfang des Projekts für Ihre Gruppe vereinbart wird. Auf dieser Basis ist dann das Pflichtenheft auszuarbeiten.

Ebenso ist die Designstudie (inkl. Vorführung des Programmes) Gegenstand des Reviews.

Im Review wird die Anforderungsanalyse an Hand der vorgelegten Dokumente und der Demo-Präsentation durchgesprochen.

Damit sich alle Seiten auf die Reviews gut vorbereiten können, ist die **rechtzeitige Vorlage** von Lastenheft und Glossar **Voraussetzung**. Beachten Sie das **Merkblatt zum Ablauf eines Reviews**.

Literaturverzeichnis

1. Balzert, Helmut; Liggesmeyer, Peter (2011): Lehrbuch der Softwaretechnik: Entwurf, Implementierung, Installation und Betrieb. 3. Aufl. Heidelberg: Spektrum Akademischer Verlag (Lehrbücher der Informatik).

Designstudie Multi-User Dungeon Server

Ergänzung zum Aufgabenblatt 3

In der Projektaufgabe soll ein textbasiertes Rollenspiel entstehen (siehe auch das Dokument Einführung.doc). In dieser Aufgabe geht es darum, erste Ideen und softwaretechnische Kompetenzen bzgl. der praktischen Realisierung solch eines Systems zu sammeln, um daraus für das später zu entwickelnde System zu lernen.

Achtung

- Diese Aufgabe ist nicht als reelles Lasten- oder Pflichtenheft zu sehen! Zur Ermittlung der Anforderungen an das Produkt, welches Sie später zu erstellen haben, müssen Sie sich unbedingt mit Ihrem Kunden zusammensetzen.
- Die Software die Sie im Rahmen dieser Aufgabe erstellen, soll durchaus den Charakter eines Wegwerf-Prototypen haben. Vermutlich haben Sie nicht die Zeit, schon etwas zu realisieren, das gut genug ist um als „Fundament“ oder Version 0 Ihres späteren Produktes zu dienen.
- Daher wird Ihre Software von den Betreuern nicht auf technische Qualität geprüft.

Aufgabe

Erstellen Sie in einer Softwarestudie eine Applikation mit folgenden Anforderungen:

/F10/ Geschäftsprozess: Spiel konfigurieren
Akteur: Dungeon Master
Der Dungeon Master kann das Spiel nach seinen Wünschen und Vorlieben gestalten. Dabei kann er verschiedene Kategorien von Inhalten für den Dungeon konfigurieren. Die Konfigurationen für die Avatare der Spieler sind: Klassen, Rassen und Equipment, wobei jede der Konfigurationen definierte Fähigkeiten des Avatars steuern. Im Dungeon verfügbare Ressourcen werden definiert. Außerdem können beliebig viele Räume angelegt und zueinander angeordnet werden, darin Objekte und nicht-Spieler Charaktere platziert und Aktionen für den Raum konfiguriert werden.

/F20/ Geschäftsprozess: Auf Plattform registrieren
Akteur: Spieler
Einem Spieler ist es möglich sich auf der Plattform zu registrieren und alle verfügbaren Spiele zu sehen.

/F30/ Geschäftsprozess: Für ein Spiel anmelden/ Aus einem Spiel abmelden
Akteur: Spieler
Ein Spieler kann sich zu einem verfügbaren Spiel anmelden und sich von einem Spiel zu dem er angemeldet ist wieder abmelden. Meldet sich ein Spieler zu einem Spiel an, so darf er den zugehörigen Dungeon mit einem selbst konfigurierten Avatar betreten. Sobald sich ein Spieler von einem Spiel abmeldet, gehen alle seine dort gesammelten Spielstände verloren.

/F40/ Geschäftsprozess: Charakter konfigurieren
Akteur: Spieler
Ein Spieler hat die Möglichkeit sich einen Charakter im Spiel, einen sogenannten Avatar, selbst zu konfigurieren.

/F50/ Geschäftsprozess: Spiel betreten/ verlassen
Akteur: Spieler
Ein Spieler kann zu jeder Zeit einen Dungeon auf dem er angemeldet ist betreten und wieder verlassen. Betritt ein Spieler den Dungeon, so wird er in einem Start-Raum platziert und bekommt Informationen zu seinem Aufenthalt und seiner Umgebung.

/F60/ Geschäftsprozess: Mit anderen Spielern chatten
Akteur: Spieler, Dungeon Master
Ein Spieler, der sich im Dungeon befindet kann in einer öffentlichen Unterhaltung mit allen im gleichen Raum befindlichen Spielern chatten. Ebenso können andere im selben Raum befindliche Spieler über eine Flüster-Funktion privat angesprochen werden, wobei dieser Chat dann für keinen anderen Spieler sichtbar ist. Der Dungeon Master kann auch jedem Raum direkt angesprochen werden und kann ebenso jeden Spieler im Dungeon unabhängig seiner Position anflüstern.

/F70/ Geschäftsprozess: Aktion durchführen
Akteur: Spieler
Befindet sich ein Spieler im Dungeon, so hat er die Möglichkeit unterschiedlichen Aktionen durchzuführen. Dazu gehören: Raum, Rauminhalt und im Raum befindliche Spieler wahrnehmen, Equipment aufnehmen, den Raum wechseln und weitere vom Dungeon Master für den Raum definierte Aktionen ausführen.

/F80/ Geschäftsprozess: Spielgeschehen steuern
Akteur: Dungeon Master
Der Dungeon Master kann zur Laufzeit des Spiels aktiv in das Spielgeschehen eingreifen und dieses steuern. Dabei kann der Dungeon Master Aktionen in jedem Raum und zu jeder Zeit ausführen wie ein Monster zu platzieren, nicht-Spieler Charaktere zu steuern oder den Schwierigkeitsgrad des Spiels zu senken oder zu erhöhen.

Hinweis

Die Softwarestudie erstellen Sie, um sich mit folgenden Konzepten und Technologien vertraut zu machen, daher sind diese auch einzusetzen:

- Software zur Verarbeitung und Übermittlung von Datenströhmen (z.B. Java Sockets, Apache Kafka, RabbitMQ, TalkJS)
- JBOSS/Tomcat

Aufgabenformat

- Ihre Lösung muss installiert und von der Website Ihres Projektes abrufbar/ausführbar sein.
- Eine kurze Dokumentation zur Handhabung der Anwendung soll verfügbar sein.