Projekttagebuch - 1 (Woche 1)

Nach der ersten Kontaktaufnahme durch den Auftraggeber[1] und unserer Zusage zum Projekt, hat er uns seine Anforderungen ausgehändigt. Wir verschaffen uns auf dieser Basis erstmal einen Überblick. Zunächst wollen wir einen Zeitplan erstellen sowie eine Aufwandsschätzung für das Projekt abgeben.

Eine Projektbeschreibung fasst aus Sicht des Auftraggebers seine Sichtweise mit den ihm wichtigen Funktionen zusammen. Diese schriftliche Zusammenfassung dient als Grundlage und wird meist durch ein Gespräch mit dem Auftraggeber ergänzt. Je nach dem wie konkret die Vorstellungen des Kunden sind, ist es auch möglich, dass kein schriftlicher Leitfaden zur Verfügung steht oder es ist möglich, dass der Kunde mit einem Lastenheft seine Vorstellungen festhält.

Da wir den Zuschlag für das Projekt unbedingt bekommen wollten, haben wir bereits ein konkretes Angebot an den Auftraggeber abgeben. Das Projekt soll in 3 Monaten abgeschlossen sein. Um im Kostenrahmen zu bleiben, können wir nur mit vier Personen à 270 Zeitstunden planen. Das macht einen Gesamtzeitaufwand von 1080 Stunden. Das wird ein knappes Ding.

Projektbeschreibung, Dateiname: projektbeschreibung.pdf

Dieses Tagebuch soll im Besonderen dem Leser helfen den Projekthergang zu erkennen. Es wird aber nicht erwartet, dass auch jede andere SWT-P Gruppe solch ein Tagebuch führt. Als hilfreich und lehrreich hat sich aber erwiesen, eine Zeit- bzw. Aufwandsschätzung zu erstellen und diese später mit dem tatsächlichen Ablauf zu vergleichen.

[1] Der Professor spielt den Kunden.

Projekttagebuch - 2 (Woche 2)

Wir haben die Projektbeschreibung erhalten. Wir haben sie gelesen und erstmal vier Arbeitsbereiche definiert:

1. Domänenanalyse
2. Benutzeroberfläche
3. Technische Infrastruktur
4. Teamorganisation

Bis zum nächsten Projekttreffen wird jeder von uns einen Teil dieser Arbeitspakete bearbeiten.

In Lehrbüchern über Softwaretechnik werden verschiedene Vorgehensmodelle für das Software Engeneering beschrieben und ihre Vor- und Nachteile erörtert. Manchmal erlebt man auch, dass Softwareentwickler sich sehr prinzipiell pro oder contra zu solchen Vorgehensmodellen äußern. Es scheint ab und zu einen „Hype“ bestimmter Methoden zu geben, der später dann vielleicht auch wieder abebbt und einem neuen „Buzzword“ weicht (sehr schön wird der „Hype“ der agilen Methoden z.B. im Buch von Ludewig/Lichter über Softwaretechnik charakterisiert).

Unsere Projektgruppe hat sich ohne lange Diskussion zu einem pragmatischen Vorgehen entschlossen, das sich in Bezug auf die zu bearbeitenden Themen an der Vorlesung SWT an der THM orientiert. Es stellt die inhaltlichen Fragen in den Vordergrund und möchte alle organisatorischen und planerischen Aspekte durch die regelmäßigen Besprechungen begleitend lösen.

Domänenanalyse

Die Domänenanalyse wird von Michael bearbeitet. Er möchte zunächst ein Datenmodell entwerfen. Dazu studiert er einige Kochbücher, um einen Überblick über die Eigenschaften von Kochrezepten zu bekommen. Weitere Informationen sammelt er in Gesprächen mit einigen Hobbyköchen. Das entworfene Datenmodell soll Teil der Spezifikation werden.

Benutzeroberfläche

Jeanette und Barbara zeichnen auf Papier eine Skizze der zukünftigen Oberfläche. Sie versuchen auch schon die Aspekte der Barrierefreiheit und der mobilen Variante mit einzubeziehen. Sie beginnen außerdem eine Liste mit den Anwendungsfällen, die sich aus der Projektbeschreibung ergeben.

Für die Beschreibung von Anwendungsfällen wird vielfach folgendes Schema (nach Alistar Cockburn) benutzt:

1. Name
2. Ziel
3. Akteure
4. Verwendete Anwendungsfälle
5. Auslöser
6. Vorbedingung / Nachbedingung
7. Standardablauf / Alternativer Ablauf

Für unsere Zwecke ist es an dieser Stelle zu komplex. Würde man sämtliche Punkte mit Informationen füllen wollen, stellt man fest, dass viele der Punkte gleich sind. Deshalb wurde eine verkürzte Variante verwendet. Für andere Projekte kann die ausführliche Variante aber die richtige sein. Oft wird eine projektspezifische Vorlage eingesetzt.

Technische Infrastruktur

Da wir alle bisher keine Erfahrungen in der Programmierung einer Webanwendung haben, aber Daniel letztens die Website des Frameworks „Wicket“ besucht hat und alles sehr vielversprechend ausgesehen hat, entscheiden wir uns für dieses Framework. Daniel arbeitet sich darin ein und versucht ein Beispiel lauffähig zu bekommen. Sobald das Beispiel funzst, wird er sein Wissen an uns weitergeben.

In unserem Musterprojekt stellen wir den Fall nach, dass das Projektteam ein Projekt mit einer bisher nicht bekannten Technologie (das Framework „Wicket“) durchführt. In dieser Situation muss man höchst agil vorgehen, da man ja nicht auf Erfahrungen mit dieser Technologie zurückgreifen kann. Deshalb hat der Projektverlauf einen explorativen Charakter.

In der Softwareindustrie ist man oft auch mit einer solchen Situation konfrontiert und genötigt Aussagen über Aufwände, technische Eigenschaften u.ä. zu Projekten zu machen, wo noch gar keine oder nur wenig Erfahrungen vorliegen. Es kann aber auch sein, dass man bereits bekannte und erprobte Techniken einsetzt. Dann ist es natürlich viel leichter den Projektverlauf zu planen und zu steuern.

Teamorganisation

Ich, Barbara, übernehme als Teamleiterin die Organisation des Teams und erstelle eine Kalendervorlage zur Planung des Projektes. Ich bringe diese zur nächsten Projektsitzung mit. Da jeder im Team des Unternehmen noch in andere Aufgaben eingebunden ist, wird jeder ca. die Hälfte seiner Wochenarbeitszeit in dieses Projekt investieren.

Projekttagebuch - 3 (Woche 4)

Wir alle haben uns intensiv mit unserem Arbeitsbereich beschäftigt. Bei einem Teamtreffen stellen wir uns gegenseitig unsere Erkenntnisse aus den verschiedenen Bereichen vor.

Nach einer Diskussion erkennen wir Defizite in unseren Dokumenten und nehmen folgende Erweiterungen an ihnen vor:

- Michael hat uns sein Datenmodell vorgestellt. Das Datenmodell ist der wichtigste Bestandteil in der Domänenanalyse. Wir diskutieren über das Datenmodell und wo wir es in die Spezifikation „einbauen“ möchten.
- Jeanette und Barbara zeigen uns die Skizze der Benutzeroberfläche und ein Beispiel für einen Anwendungsfall. Uns fällt auf, dass sie vergessen haben, die mobile Variante der Anwendung zu berücksichtigen und wir diskutieren, wie man sie gestalten könnte.
- Außerdem sprechen wir über die Liste der Anwendungsfälle und wir nehmen zwei weitere Anwendungsfälle auf.
- Zusätzlich zu den schon diskutierten Arbeitsschritten haben wir entschieden, uns an eine Empfehlung unseres Professors des letzten Semesters zu halten. Er stellte die *Kurzanleitung Projektdokumentation* vor. Dort wird nach einem allgemeinen Teil ein „Rezept für das SWT-Praktikum“ vorgestellt. Michael schreibt zusätzlich zu der Domänenanalyse ein Exposé.

Barbara stellt die Zeitplanung den anderen vor. Die orangen Balken im Zeitplan beschreiben den Anteil an dieser Tätigkeit in der jeweiligen Woche. Wir sind alle damit einverstanden.

Aufgabenname	Beginn	Abschluss	Dauer
Analyse	Woche 1	Woche 5	4 Wochen
Entwurf	Woche 3	Woche 7	4 Wochen
Implementierung	Woche 6	Woche 12	6 Wochen
Systemtest	Woche 10	Woche 13	3 Wochen
Auslieferung	Woche 13	Woche 13	1 Woche

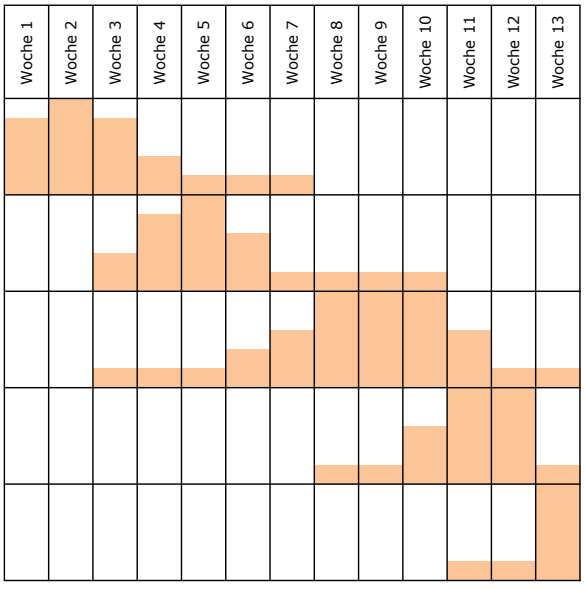

Abbildung 1: Zeitplanung

Bei der vorgestellten Zeitplanung sind zwei Punkte zu beachten: Diese Zeitplanung wurde von einer unerfahrenen Gruppe erstellt. Keinesfalls darf sich zu strikt an den Zeitplan gehalten werden. Die unterschiedlichen Tätigkeiten werden pragmatisch abgearbeitet. Zum Beispiel werden während des Entwurfs sicherlich schon Teile der Implementierung ausgeführt. Denn ohne die Konzepte hinter Wicket zu kennen, ist es unmöglich den Entwurf zu gestalten.

Häufig werden Zeitpläne mit Meilensteinen strukturiert. Damit kann man leicht den Fortschritt eines Projektes überblicken und kontrollieren. Wir wollten der Gruppe keine Vorgabe machen, Meilensteine verwenden zu müssen. Wir wissen, dass viele Dozenten für SWT-Projekte auch Meilensteine vorgeben, was dem Projektablauf von vornherein ein Gerüst gibt.

Daniel berichtete uns über die Funktionsweise von Apache Wicket und zeigte uns einige Seiten einer Wicket-Anwendung, damit wir sehen konnten, wie die Entwicklung mit dem Framework aussehen könnte. Die wichtigsten Punkte über Wicket wollen wir hier im Tagebuch festhalten.

Apache Wicket

Eine Wicket-Anwendung besteht aus einem Komponenten-Baum und Markup (in der Regel XHTML). Die Markup-Dateien enthalten - im Gegensatz zu vielen anderen Frameworks - keine Logik. Mittels Wicket-spezifischen XML-Attributen werden Elemente des Markups „markiert“. Die Komponenten beziehen sich auf diese Markierungen. Des Weiteren arbeiten die Komponenten in Wicket mit Modellen und Ereignissen, so dass sich die Entwicklung mit Wicket ähnlich der mit Swing gestaltet.

Component Komponenten sind das zentrale Konzept in Wicket. Sie enthalten in der Regel andere Komponenten und füllen Elemente im Markup mit Daten. Folgende werden wir häufig verwenden:

- *Page*: Eine spezielle Komponente, die eine gesamte Seite repräsentiert
- *Panel*: Ein Container für andere Komponenten
- *Label*: Zeigt einen Text auf einer Seite an
- *Form*: Repräsentiert ein Formular und enthält *FormComponent*s
- *TextField*, *Button*, *ListChoice*, *TextArea*: Einige häufig benötigte *FormComponent*s
- *ListView*, *PageableListView*: Komponenten zur Darstellung von Listen
- *Link*, *ExternalLink*: Link zu anderen *Page*s der Anwendung bzw. externen Resourcen

Selbstverständlich können auch eigene Komponenten entwickelt werden. In der Regel erweitert man einen *Panel* und gibt der Komponente eine eigene Markup-Datei.

Model Modelle halten die Daten der Komponenten. Jede Komponente besitzt daher ein Modell und bietet Methoden, um an dieses zu gelangen bzw. es auszutauschen. Modelle sind eine Implementierung der Schnittstelle *IModel*, welche folgende Methoden hat:

- *getObject*: Liefert das Daten-Objekt in Form von *java.lang.Object*
- *setObject*: Setzt das Daten-Objekt
- *detach*: Kann Resourcen freigeben, die nicht mehr benötigt werden. Wicket ruft diese Methode am Ende jeder Anfrage bei allen beteiligten Komponenten und Modellen auf.
- *getNestedModel*: Modelle können geschachtelt werden. Z.B. kann ein Formular ein Modell „Person“ besitzen und das enthaltene Textfeld ein Modell mit dem Namen dieser Person. Die Methode *getNestedModel* liefert das übergeordnete Modell zurück.

Wichtige *IModel*-Implementierungen sind *PropertyModel*, mit welchem man Eigenschaften eines Objektes anhand eines Ausdrucks an ein Modell binden kann und *LoadableDetachableModel*, welches eine bequeme Möglichkeit bietet Daten „on demand“ zu laden und während der Detach-Phase von Komponenten und Modellen zu verwerfen.

Behaviour Neben Komponenten und Modellen existieren Behaviours, um Komponenten mit Zusatzfunktionalität erweitern zu können ohne eine eigene Komponente erstellen zu müssen. Beispielsweise können so „DatePicker“ zu Textfeldern hinzugefügt werden oder Bestätigungspopups vor das Absenden eines Formulars geschaltet werden.

Event Auf den ersten Blick wirkt es in einer Webanwendung vielleicht ungewöhnlich, aber in Wicket reagiert man auf Ereignisse genau wie man es in Desktopanwendungen tun würde. Typischerweise wird dies bei Buttons, Links und ähnlichem verwendet.

AJAX Wicket bringt seine eigene AJAX-Library mit, die zwar von der Funktionalität her etwas reduzierter ist als beispielsweise Dojo, jedoch speziell auf Wicket, d.h. das Framework auf der Serverseite, zugeschnitten ist. Darüber hinaus bietet Wicket einige vorgefertigte Komponenten wie *AjaxLink* oder *AutoCompleteTextField*, die einem die Arbeit erleichtern. Das Grundprinzip ist stets, dass man auf ein Ereignis reagiert und dabei eine *AjaxRequestTarget*-Instanz erhält, welcher man die Komponenten, die sich geändert haben, mitteilt. Außerdem gibt es einige vorgefertige Ajax-Behaviours.

Projekttagebuch - 4 (Woche 5)

Michael hat das Exposé mit zur Teamsitzung mitgebracht. Wir lesen es gemeinsam und korrigieren einige Tippfehler. Unter dem Punkt „Infrastruktur“ hat Michael das Datenbankmanagementsystem PostgreSQL und das Persistenzframework Hibernate als zu verwendende Techniken festgelegt. Zunächst herrscht etwas Unmut unter den Teammitgliedern. Michael begründet seine Entscheidung aber zufriedenstellend:

PostgreSQL wurde in der Vorlesung verwendet und daher haben wir bereits Erfahrung mit diesem DBMS. Hibernate ist zwar bisher nur vom Namen und Konzept des objekt-relationalen Mappings her bekannt, aber eine Recherche ergab, dass es quasi Standard bei Anwendungen dieser Art ist. Das Team ist zufrieden und stimmt der technischen Infrastruktur zu.

Die letzte Woche haben wir gemeinsam am Entwurf gearbeitet. Nachdem sich besonders Daniel in die Funktionsweise von Apache Wicket eingearbeitet hat, gibt er die Grobstruktur der Anwendung vor.

Das Grundgerüst wird durch eine HTML-Seite abgebildet. In dieser befinden sich die auf jeder Seite wiederkehrenden Elemente. Die wechselnden Inhalte werden durch weitere Seiten dargestellt. Auch zu ihnen existiert jeweils eine HTML-Seite. Des Weiteren gibt es zu jeder HTML-Seite eine Repräsentation in Java. Hier werden die Datenfelder mit Daten belegt und die Navigationslogik hinterlegt.

Daniel erklärt viel über Wicket und die Konzepte dahinter. Er hat in den letzten Wochen großes Wissen darin angesammelt. Jetzt kristallisiert sich heraus wie wir Wicket - dies findet sich im Entwurf wieder - einsetzen werden.

Exposé, Dateiname: expose.pdf

Projekttagebuch - 5 (Woche 6)

Die Entwurfsphase neigt sich dem Ende zu. Wir haben viel geschafft und sind mit unserem Entwurf zufrieden. Nächste Woche startet laut unserem Zeitplan die Implementierungsphase. Wir haben folgende Aufgabenteilung vorgenommen:

- Die Umsetzung des Datenmodells mit Java, Hibernate und SQL übernimmt Jeanette.
- Layout und Design wird von Michael umgesetzt.
- Die restlichen Implementierungsaufgaben werden von Daniel und mir per Pair-Programming umgesetzt.

Nach längerer Diskussion über die Spezifikation nehmen wir einige kleinere Änderungen vor und sehen die Spezifikation als fertig an. Leider gibt Barbara als Teamleiterin eine „Hiobsbotschaft“ bekannt.

Der Auftraggeber hat uns eine weitere Anforderung hereingereicht: Jeweils für 7 Tage soll automatisch ein Essensplan erstellt werden. Wir überarbeiten daraufhin nochmals unsere Spezifikation.

Es ist nicht ungewöhnlich, dass in wirklichen Projekten weitere Anforderungen während der Planungs- und/oder Implementierungsphase umgesetzt werden sollen. Um diese zu vermeiden bzw. zu minimieren zahlt sich eine gründliche Anforderungs- und Domänenanalyse aus. Wenn trotzdem Änderungen in einer späteren Phase der Entwicklung notwendig werden, ist es wichtig mit dem Kunden eine Lösung zu finden, die sowohl die Kosten- als auch die Zeitplanung berücksichtigen.

Wir haben den Eindruck, dass wir allmählich die Konzepte, wie man mit Wicket eine Webanwendung erstellt im Prinzip verstanden haben. Wir wollen jedoch die Dokumentation des Entwurfs noch nicht abschließen, weil wir davon ausgehen, dass während der Implementierung noch neue Gesichtspunkte auftreten.

Unserer Projektgruppe hat noch nicht viel Erfahrung in der Entwicklung von Webanwendungen und muss deshalb damit rechnen, dass in der Implementierung Änderungen am Entwurf möglich und sogar wahrscheinlich sein werden. Deshalb ist es vernünftig, dass die Projektgruppe den Entwurf noch nicht endgültig abschließt. Man sollte jedoch bedenken, dass man Entwurfsdokumente nicht „endlos“ „nachziehen“ kann. Meist wird deshalb in vielen Projekten der Entwurf zu einem bestimmten Zeitpunkt als Bauplan abgeschlossen, auch wenn den Beteiligten bewusst ist, dass beim tatsächlichen Bau, also der Implementierung, noch Änderungen auftreten werden.

Für die nächsten Schritte haben wir folgende Arbeitsteilung vorgenommen:

- Barbara wird die Spezifikation abschließen.
- Jeannette will das Datenmodell fertigstellen.
- Das Layout bearbeitet Michael.
- Die Implementierung der Java-Klassen wird per Pair Programming von Daniel und Jeanette übernommen.

Projekttagebuch - 6 (Woche 8)

Barbara hat die Spezifikation abgeschlossen und stellt sie den anderen vor. Wir überprüfen inwiefern die Anwendungsfälle diese gewünschten Funktionen abdecken. Wir stellen gemeinsam fest, dass die Spezifikation so in Ordnung und damit fertig ist.

Spezifikationen enthalten in der Regel über die Anwendungsfälle hinaus Listen der benötigten Funktionalität die vollständig und konsistent sein müssen. Die Überprüfung des erstellten Produkts, ob es die gewünschte Funktionalität erfüllt, wird dann anhand der Listen durchgeführt.

Im Beispiel unseres Musterprojektes sind die funktionalen Anforderungen überschaubar und die Skizzen der Oberfläche sowie die dokumentierten Anwendungsfälle geben einen Überblick über die benötigte Funktionalität. Für ein Projekt dieser Größe kann man dies als ausreichend ansehen.

In der Spezifikation sind keine qualitativen Anforderungen wie Performance, Sicherheit, Ergonomie, Zuverlässigkeit, Wartbarkeit und Erweiterbarkeit betrachtet. Nach einer Diskussion kommen wir zum Schluss, dass wir uns eigentlich nur auf die Eigenschaften der verwendeten Infrastruktur (DBMS, Webframework, Servlet Container) verlassen. Zeitlich ist es schwierig diese Gesichtspunkte gründlich zu berücksichtigen. Wir wollen aber während der Implementierung ein Auge darauf haben.

Die in der Diskussion der Projektgruppe genannten qualitativen Eigenschaften sind natürlich eigentlich der Kern des Themas „Softwaretechnik“. Deshalb sollte diese Diskussion unbedingt geführt werden.

Andererseits ist uns als Betreuer von Softwaretechnik-Projekten an unserer Hochschule durchaus bewusst, dass die Studierenden in großem Maße auf die Qualitätsmerkmale der von ihnen eingesetzten Technologien angewiesen sind und kaum in der Lage sind, diese in der Tiefe auszuloten. Wichtig ist uns, dass diese Gesichtspunkte im Auge behalten werden.

In der Spezifikation unserer Projektgruppe werden die Anwendungsfälle nur mittels Text beschrieben und kein Anwendungsfalldiagramm als Überblick und zur Strukturierung des Zusammenhangs der Anwendungsfälle verwendet. In einem Projekt mit reichhaltigeren Anwendungsfällen wird oft das Anwendungsfalldiagramm eingesetzt.

Spezifikation, Dateiname: spezifikation.pdf

Die letzte Woche haben wir uns außerdem mit der Implementierung beschäftigt.

Jeanette hat die Aufgabe der Datenmodellierung übernommen und das Klassendiagramm in Java-Code umgesetzt. Das SQL-Skript zur Erstellung der Datenbank ist nun auch fast fertig. Im Moment fügt sie die Hibernate-Annotationen in den Java-Code ein. Um das Datenmodell während der Entwicklungsphase auf Richtigkeit zu testen, hat sie sich eine Klasse `DataTest` geschrieben, die jeweils Entity-Objekte anlegt und diese mit Hilfe von Hibernate persistiert. Jeanette bereitet zum nächsten Projekttreffen ein ER-Diagramm der Datenbank vor. Daran möchte sie ihren Mitstreitern die Datenbank erläutern und endgültig abstimmen. Besonders die Navigierbarkeit zwischen den einzelnen Objekten möchte sie besprechen.

Hibernate kann man grundsätzlich auf zwei Arten verwenden:

- Das Datenbankschema kann aus den annotierten Java-Entity-Klassen generiert werden, oder
- aus einem bestehenden Datenbankschema werden die Java-Entity-Klassen generiert.

Die Projektgruppe hat sich offenbar dafür entschieden, beides selbst zu schreiben und für die Übereinstimmung zu sorgen - eine schöne Übung zum Verständnis des objekt-relationalen Mappings mit der Java Persistence API.

Michael hat das Grundlayout und das Design der Anwendung festgelegt. Als Hauptfarbe wird grün verwendet. Links oben befindet sich das Logo der Tomate und Salat AG. Im Kopfbereich befindet sich zentral die Suche und auf der rechten Seite der Login-Bereich. Auf der rechten Seite weiter unten wird Platz für 5 wechselnde Rezepte und die Tag-Cloud sein. Die anderen Teammitglieder finden seinen Vorschlag gut und beschließen den Aufbau so beizubehalten.

Daniel und ich haben uns vor allem mit der Umsetzung des skizzierten Grobentwurfs befasst. Die Implementierung zeigt, dass wir den Aufbau und die sich daraus ergebende Verwendung von Wicket verstanden haben. Allein der Einsatz der Models in Wicket war uns zunächst nicht klar bzw. wir haben bei unseren eigenen Komponenten die Vorteile dieses Konzeptes nicht genutzt. Denn wir haben die Daten einer Komponente ohne Verwendung eines Models bei deren Erstellung übergeben. Also z.B. bei einer Komponente zur Darstellung einer Liste von Rezepten wurden diese dem Konstruktor einfach als `java.util.List<Recipe>` übergeben. Wenn der Konstruktor anstatt dessen ein `IModel<java.util.List<Recipe>>` erwartet, hat der Verwender der Komponente viel mehr Möglichkeiten, da er die Liste dann auf ganz unterschiedliche Weise übergeben kann bzw. teilweise gar nicht selbst übergeben muss - eben je nach dem, welche `IModel`-Implementierung er verwendet. Diese Vorteile, die das Wicket-Framework bietet, kann man bei eigenen Komponenten natürlich nur nutzen, wenn diese sich sozusagen in das Wicket-Universum eingliedern, was wir nun nachgeholt haben.

Alle im Team haben im Kopf, dass wir die Anwendung gut dokumentieren müssen. Wir haben uns entschieden möglichst zeitnah (z.B. wenn eine neue Klasse oder darin enthaltene Methoden erstellt werden) die Schnittstellendokumentation zu schreiben. Dieses Verfahren soll helfen, dass wir eine möglichst lückenlose API-Doku bekommen.

Die Dokumentation ist in Softwareprojekten meist eine unbeliebte Tätigkeit und wird gern vernachlässigt. Hinzu kommt, dass gerade gegen Ende eines Projektes die meisten Teilnehmer viel Zeit für die unaufhaltsam näherrückende Auslieferung aufwenden (müssen). Die Vernachlässigung der Dokumentation rächt sich aber spätestens nach einem halben Jahr oder wenn die Entwickler (zum Teil) gewechselt haben. Bei der Dokumentation und besonders bei der Codekommentierung geht es nicht um das Erstellen möglichst vieler und umfangreicher Dokumente, sondern vor allem um das Konservieren des Implementationsgedankens. In diesem Projekt wurde nach den Java-Codierrichtlinien des Fachbereichs MNI kommentiert.

Als weiterer Aufgabenbereich ist die Benutzerdokumentation hinzugekommen. Der Auftraggeber hat den Wunsch nach einem Handbuch geäußert. Wir konnten aber geschickt argumentieren und werden statt eines statischen Dokuments ein leichter anpassbares Online-Handbuch erstellen, das direkt in die Anwendung integriert ist. Dieses wird sich an den Anwendungsfällen orientieren und

diese jeweils für den Benutzer durchspielen.

Projekttagebuch - 7 (Woche 9)

Die Arbeiten an der Implementierung schreiten voran. Wir sind zwar noch in der Hauptphase der Implementierung, trotzdem hinken wir dem Zeitplan leider etwas hinterher, da Michael schon dreimal nicht zu den Teamtreffen erschienen ist und auch mit der Umsetzung des Layouts im Rückstand ist. Beim letzten Teamtreffen habe wir das mit ihm thematisiert und er hat uns zugesagt in Zukunft wieder voll und ganz hinter der Sache zu stehen.

Abweichungen vom Zeitplan sind leider häufig die Regel. Die ist zum Teil darin begründet, dass der Zeitplan vieler Projekte zu knapp kalkuliert wird und dadurch Abweichungen vorprogrammiert sind. Manchmal treten aber auch Unstimmigkeiten im Team auf, die zunächst gelöst oder zumindest akzeptiert werden müssen. Auch Verzögerungen, die durch den Auftraggeber verursacht werden, sind möglich. In dieser Situation besteht immer das Problem, dass auch diese Verzögerungen auf das Projektmanagement zurückfallen.

Jeannette berichtet vom Datenbankschema-Diagramm und macht darauf aufmerksam, dass der Entwurf in eine finale Version gebracht werden sollte. Barbara bittet alle ihre Teile fertig zu machen. Sie wird sie in ein Dokument integrieren - für das nächste Treffen. Dafür wird Michael bei der Implementierung mithelfen.

Projekttagebuch - 8 (Woche 10)

Wir haben den Entwurf diskutiert und verabschiedet. Unabhängig davon sind folgende Punkte aufgefallen:

- In Fehlermeldungen, die sich auf Feldnamen beziehen, tauchen noch die englischen Variablennamen auf.
- Die zusätzliche Anforderung „7-Tage-Planer“ des Kunden wurde noch nicht umgesetzt[2]

Entwurf, Dateiname: entwurf.pdf

Ich habe mich bereit erklärt, die noch offenen Punkte bis zum nächsten Teamtreffen zu bearbeiten.

Als Testansatz verfolgen wir den Use-Case-zentrierten Ansatz. Dazu wollen wir zunächst die Testumgebung mit verschiedene Ausgangssituationen definieren und anschließend verschiedene Szenarien anhand der Anwendungsfälle durchspielen. Die Konzeption davon wird von Jeanette, Daniel und Michael durchgeführt.

[2] Dem aufmerksamen Leser wird aufgefallen sein, dass in der Spezifikation auf Seite 13 die UseCase-Berechnung fehlt. Hier ist der Grund.

Projekttagebuch - 9 (Woche 12)

Die Zeit seit dem letzten Treffen wurde vor allem für die Implementierung der Tests genutzt. Da dies überwiegend abgeschlossen ist, möchten wir hier das gewählte Konzept festhalten.

Wicket verfügt über ein integriertes Testframework. Dazu bietet Wicket die Klasse `WicketTester` an, die es beispielsweise erlaubt, den Aufruf von Pages oder einzelner Panels zu simulieren und anschließend auf Fehlermeldungen, Sichtbarkeit bestimmter Komponenten, Enthaltensein von Strings im gerendertem Markup und ähnlichem zu testen. Des Weiteren stellt Wicket die Klasse `FormTester` bereit, mit dessen Hilfe Formulare automatisiert ausgefüllt und abgesendet werden können.

Um die Entwicklung neuer Test-Cases zu vereinfachen, haben wir die Klasse `WicketTestCase` entwickelt, welche von der JUnit-Klasse `TestCase` ableitet und durch Implementierung von `setUp` und `tearDown` dafür sorgt, dass jeder Test unabhängig von den vorherigen abläuft. Dies wird erreicht indem die verwendete Test-Datenbank nach jedem Test geleert wird. Zusätzlich bietet `WicketTestCase` Bequemlichkeitsmethoden für häufig benötigte Aufgaben wie das Anlegen von Test-Daten, die Anmeldung eines Benutzers oder das Aufrufen eines bestimmtes Rezeptes.

Wir verwenden im Wesentlichen zwei Arten von Tests. Zum einen Tests von Use-Cases und zum anderen Tests einzelner Komponenten. Bei Ersteren wird nach dem Blackbox-Verfahren vorgegangen, da lediglich die Spezifikation der Use-Cases dafür herangezogen wird. Um wichtige Pages und Panels ausführlicher zu testen, d.h. auch von den Use-Cases nicht berücksichtigte Situationen abzudecken, wurden zusätzliche Tests für diese entwickelt. In diesem Fall handelt es sich eher um Whitebox-Tests. Typischerweise wird dabei so vorgegangen, dass eine Page gestartet wird, ein Link darauf angeklickt wird, ein Formular ausgefüllt wird und schließlich abgesendet wird. Bei jedem der Schritte wird sichergestellt, dass die richtige Page gerendert wurde und keine Fehler aufgetreten sind. Nach Ausführung einer Aktion (z.B. Login, Registrierung, Anlegen eines Rezeptes, ...) wird mittels der Datenschicht verifiziert, ob die Aktion erfolgreich ausgeführt wurde, d.h. ob das gewünschte Ergebnis erzielt wurde.

Projekttagebuch - 10 (Woche 13)

Die Auslieferung an den Kunden war erfolgreich. Der Abnahmetest verlief ohne größere Beanstandungen. Die optische Umsetzung wurde besonders gelobt. Jedoch stellte der Kunde nach kurzer Zeit einen kleinen Fehler fest. Bei der Registrierung wurde die maximale Länge des Benutzernamens nicht geprüft, so dass es bei zu langen Namen zu einem Laufzeitfehler kam. Dies wurde sofort behoben und es wurde auch ein entsprechender Unit-Test hinzugefügt. Seit dem läuft alles zur vollen Zufriedenheit des Kunden. Der Zeitaufwand, den wir fürs Testen aufgebracht haben, hat sich also gelohnt, da wir durch unser systematisches Vorgehen die meisten Fehler vor der Auslieferung ausmerzen konnten.

Autoren: Nils Asmussen, Nadja Krümmel, Burkhardt Renz, Projektgruppe Softwaretechnik.