Universität Zürich[UZH]

Institut für Informatik

Software Engineering
Herbstsemester 2016

Übung 4

Sprint 1 und Aufgabenbeschreibung für Sprint 2

Allgemeine Informationen

Ausgabe	Abgabe	Besprechung	Maximalpunktzahl
Dienstag, 1.11.2016	Montag, 14.11.2016, 23:59 Uhr	Sprint Meeting, Dienstag, 22.11.2016	20

Formales

Ihre Abgabe besteht aus Ihrem Lösungsdokument und Ihrem Source Code. Das Dokument benennen Sie wie folgt: **Ex4 Gruppe [Nummer].pdf**. Ihren Source Code geben Sie entweder als GitHub Link oder als zip-Datei ab. Die Abgabe der Lösungen erfolgt via OLAT. Archivieren Sie das Lösungsdokument und den Source Code in einer zip-Datei und laden diese auf OLAT hoch. Die zip-Datei soll nach den oben erwähnten Konventionen benannt werden. Falls der Source Code die erlaubte Uploadgrösse von OLAT überschreitet, soll lediglich der *src* Folder, ohne GWT Zusatz, eingesendet werden.

Sie bearbeiten diese Übung in Ihrem Projektteam. Bestimmen Sie ein Teammitglied, welches am Schluss die gemeinsam erarbeiteten Ergebnisse auf OLAT hochlädt. Verspätete Abgaben werden nicht korrigiert.

Alle Studierenden des Projektteams müssen an der Bearbeitung beteiligt sein und über alle Teile der Lösung Auskunft geben können.

Überblick und Ziele

Hintergrund

In Übung 3 haben Sie die Sprint-Auftragsliste *(Sprint Backlog)* für Sprint 1 erstellt. Sie haben die Benutzergeschichten für Sprint 1 auf einzelne Aufgaben bzw. Arbeitseinheiten *(Tasks)* heruntergebrochen und diese den einzelnen Gruppenmitgliedern zugeteilt.

Die Ziele dieser Übung sind

- eine lauffähige, getestete Applikation und
- die Aufgabenbeschreibung für Sprint 2.

Dazu implementieren Sie zuerst die Applikation gemäss den erstellten Arbeitseinheiten (*Tasks*) aus Übung 3 und planen dann den zweiten Sprint. Die Planung von Sprint 2 beinhaltet die Erstellung einer entsprechenden Aufgabenvereilung *(Work breakdown)*.

Führen Sie die Planung von Sprint 2 erst **nach** der Beendigung von Sprint 1 aus, damit Sie nicht gelöste Probleme aus Sprint 1 (zum Beispiel Fehler in der Implementierung oder wegen Zeitmangel nicht implementierte Arbeitseinheiten) bei der neuen Aufgabenverteilung berücksichtigen können.

Lernziele

Sie lernen in dieser Übung die Wichtigkeit einer genauen Aufgabenplanung kennen. Weiter lernen Sie mit Anpassungen in Ihrem Projektplan umzugehen. Sie wissen nach dieser Übung, wie Fehler im Code erkannt werden, wie man sie dokumentiert und wie Sie vorgehen können, um sie zu beheben.

Aufgabenstellung Teil 1: Durchführung von Sprint 1

Alle Teilaufgaben basieren auf der Anforderungsspezifikation, welche Sie bereits in Übung 2 erstellt haben. Zusätzlich können Sie auf die Informationen in der Aufgabenstellung von Übung 2 sowie auf Ihre Notizen aus der Besprechung mit den Auftraggebern zurückgreifen. Wenn Sie Unklarheiten haben, dann benutzen Sie das Forum in OLAT.

Wie schon in der Vorlesung erwähnt, verwenden wir in unserem Übungsprojekt einen hybriden Entwicklungsprozess. Die vergleichsweise ausführliche Dokumentation der Anforderungen sowie der Architektur zu Beginn der Entwicklung ist eher typisch für ein ergebnisorientiertes Phasenmodell. Die Realisierung in Inkrementen fester Dauer entspricht dagegen einer agilen Vorgehensweise im Sinne von Scrum.

Der Ablauf dieser Übung sowie der Übungen 5 und 6 lehnt sich an Scrum an. Wir verwenden jedoch wesentliche Elemente von Scrum (z.B. Scrum-Master, Daily Scrum) nicht, weil diese sich im gegebenen Übungskontext (viele Übungsgruppen, begrenzte Arbeitskapazitäten von Teams und Betreuern) nicht sinnvoll realisieren lassen.

In der Praxis existieren die Rollen von Auftraggebern und Produkteignern getrennt voneinander. Im Rahmen dieses Übungsprojektes ist diese Unterscheidung nicht möglich. Wenn wir daher im Folgenden vom Produkteigner sprechen, so entspricht dies dem, was wir in den Übungen 2 und 3 als Auftraggeber bezeichnet haben.

Wir haben diese hybride Vorgehensweise bewusst gewählt, um den im Rahmen eines Übungsprojekts erzielbaren Lernerfolg zu optimieren.

Teilaufgabe 1.1: Das Ziel von Sprint 1 **1 Punkte**

Legen Sie auf der Basis der bisher erhobenen Anforderungen und der Planung von Sprint 1 das Ziel Ihres Sprints fest. Beschreiben Sie kurz (maximal 50 Worte), was in diesem Sprint erreicht werden soll.

Das Sprint-Ziel wird verwendet, um Aussenstehende über den Sprint zu informieren. Es gibt Interesseneigner *(Stakeholders)*, die wissen wollen, woran das Team gerade arbeitet, aber keine Details über die einzelnen Benutzergeschichten in der Produkt-Auftragsliste wissen müssen oder wollen.

Beispiel zur Veranschaulichung: Sprint-Ziel für die Implementierung eines Einkaufswagens in einem Onlineshop: „*Implementieren Sie die grundlegende Funktionalität eines virtuellen Einkaufswagens. Diese beinhaltet das Hinzufügen und Entfernen von gekaufter Ware, sowie das Aktualisieren des virtuellen Einkaufswagens*“.

Der Erfolg des Sprints wird später in der Sprint Review Sitzung anhand des Sprint-Ziels beurteilt (und nicht anhand der einzelnen Benutzergeschichten).

Die Lösung dieser Teilaufgabe besteht aus einem Satz, welcher das Ziel des Sprints beschreibt.

Teilaufgabe 1.2: Durchführung von Sprint 1 **8 Punkte**

Das Hauptziel dieser Teilaufgabe ist es, ein funktionierendes Pilotsystem am Ende von Sprint 1 zu haben. Dabei sollen Sie sich auf die zum jetzigen Zeitpunkt kritischen Benutzergeschichten konzentrieren. Nach der Aufgabenplanung für Sprint 1 in Übung 3 sollte jedes Teammitglied ungefähr dieselbe Auslastung haben. Falls Sie nach dem Erhalt des Feedbacks am 8.11.2016 das Gefühl haben, die Aufgaben neu verteilen oder anpassen zu müssen, dürfen Sie dies tun.

Vergessen Sie nicht, die Änderungen zu dokumentieren und die Dokumentation bei der Abgabe dieser Übung ebenfalls mitzuliefern.

Ihre Aufwandschätzung wird sich im Verlaufe des Projektes wahrscheinlich noch ändern. Es ist nicht auszuschliessen, dass die Implementierung einzelner Benutzergeschichten sich als aufwändiger herausstellt als Sie diese geschätzt haben. Dies kann zur Folge haben, dass Sie nicht alle der für diesen Sprint ausgewählten Benutzergeschichten implementieren können. Stellen Sie in jedem Fall sicher, dass Sie am Ende des Sprints mindestens die kritischen (d.h. die für eine funktionierende Applikation zwingend notwendigen) Benutzergeschichten implementiert haben.

Teilen Sie die Arbeit entsprechend Ihrer aktualisierten Aufgabenverteilungsliste (*Work Breakdown*) auf die Teammitglieder auf. Führen Sie die Ihnen zugewiesenen Aufgaben aus, indem Sie die entsprechende Software entwerfen, codieren und gemäss Teilaufgabe 1.3 testen.

Integrieren Sie die getesteten Module mit dem Google Web Toolkit als Basis und erzeugen Sie ein Pilotsystem, welches als Web-Applikation funktionsfähig ist und auf das über einen Browser zugegriffen werden kann.

Die Lösung dieser Teilaufgabe besteht aus einem funktionierenden, gemäss Teilaufgabe 1.3 getesteten Pilotsystem Ihrer geplanten Applikation. In Ihrer Übungsabgabe müssen Sie die URL Ihrer App angeben, damit wir die Funktionsfähigkeit der App überprüfen können. Zusammen mit Ihrem Team werden Sie dieses Pilotsystem am 22.11.2016 dem Produkteigner präsentieren.

Hinweise

- Die Arbeiten an den Teilaufgaben 1.2, 1.3a, 1.3b sollten zeitlich verzahnt und nicht sequenziell nacheinander ausgeführt werden.
- Planen Sie die Implementierung so, dass Ihrem Team noch Arbeitskapazität für den Systemtest, sowie für die Suche und die Behebung von Fehlern (Teilaufgabe 1.3c) verbleibt.
- Es werden keine Punkte abgezogen, falls Sie in dieser Phase Benutzergeschichten oder Arbeitseinheiten auf einen späteren Sprint verschieben müssen, solange diese Änderungen wie folgt *dokumentiert* sind: was genau mehr Aufwand als erwartet verursacht hat, welche Benutzergeschichten deshalb nicht implementiert werden konnten und weshalb Sie sich entschieden haben, genau diese Benutzergeschichten nicht zu implementieren.
- Ein nicht getestetes, sowie ein nicht lauffähiges Pilotsystem führen zu Punkteabzügen.

Teilaufgabe 1.3: Testen **7 Punkte**

In der agilen Softwareentwicklung ist das Testen ein Kernbestandteil jedes Sprints.

a) Komponententest (3 Punkte)

Schreiben Sie JUnit-Tests, welche alle Methoden überdecken, die Sie geschrieben haben, d.h. mindestens einen Testfall pro Methode. Falls eine Methode Ausnahmen (exceptions) wirft, müssen Sie auch diese mit Testfällen abdecken. Wenn sich eine Methode nicht sinnvoll mit JUnit testen lässt (beispielsweise, weil sie grafische Ausgaben auf der Benutzerschnittstelle erzeugt), dann müssen Sie begründen, weshalb Sie für die betreffende Methode keine JUnit Tests erstellt haben. Diese Methoden müssen durch Testfälle im Systemtest (vgl. Teilaufgabe 1.3b) abgedeckt werden. In der agilen Software-Entwicklung schreiben die Entwickler die Modultests (Unit-Tests) entweder vor dem Code oder parallel zum Code und testen auch selbst.

Entwickeln Sie den Code (vgl. Teilaufgabe 1.2) und die zugehörigen JUnit Tests parallel. Wer eine Methode codiert, schreibt auch die zugehörigen JUnit Tests und ist dafür verantwortlich, dass diese Tests ohne Befunde durchlaufen. Die Entwicklung einer Methode oder Klasse ist erst abgeschlossen, wenn die zugehörigen JUnit Tests keine Fehler mehr anzeigen!

Falls Sie sich mit JUnit Tests nicht auskennen, finden Sie die benötigten Informationen im Tutorial "Testing_Tutorial.pdf", welches Sie von der Kurswebseite oder über OLAT herunterladen können.

b) Testvorschrift für den Systemtest (2 Punkte)

JUnit-Tests sind Modultests. Sie können damit beispielsweise keine Fehler finden, welche durch eine fehlerhafte Zusammenarbeit der in Ihrem System verwendeten Module verursacht werden. Solche Fehler zeigen sich typischerweise durch ein unkorrektes Systemverhalten (beispielsweise bei der Abarbeitung eines Anwendungsfalls) oder durch Fehler in der Benutzerschnittstelle. Sie müssen daher zusätzlich zu den automatisierten JUnit-Tests auch manuell testen.

Schreiben Sie eine Testvorschrift für einen manuellen Systemtest, mit der Sie das Verhalten Ihres Systems und die Benutzerschnittstelle testen. Orientieren Sie sich bei der Auswahl der Testfälle an den Abnahmekriterien, die Sie für die in Sprint 1 verwendeten Benutzergeschichten festgelegt haben.

c) Durchführung des Systemtests (2 Punkte)

Führen Sie den Systemtest mit der in Teilaufgabe 1.3b erstellten Testvorschrift durch und dokumentieren Sie das Resultat. Wenn der Test Fehler findet, suchen Sie die Fehlerquelle (d.h. die fehlerverursachenden Defekte) und beheben Sie diese. Danach ist der Systemtest zu wiederholen. Das Ganze wiederholen Sie so lange, bis Sie mit Ihrer Testvorschrift keine Fehler mehr finden. Wenn Ihnen im Rahmen der für den Sprint verfügbaren Arbeitskapazität keine Zeit mehr für die Defektsuche und -behebung bleibt, erstellen Sie für jeden nicht behobenen Fehler einen Fehlerbericht *(Bug Report)*. Dieser besteht aus folgenden Feldern: Fehler-ID; Kurzbeschreibung des Fehlers; Nummer des Testfalls, bei dem der Fehler auftritt; Verfasser; Priorität; Aufwand. Die Felder für Priorität und Aufwand bleiben vorerst leer.

Die Lösung von Teilaufgabe 1.3 besteht aus den von Ihnen geschriebenen und ausgeführten JUnit Tests, den Begründungen für nicht erstellte JUnit Tests, der Systemtestvorschrift mit den eingetragenen Befunden der letzten Durchführung des Systemtests sowie gegebenenfalls den Fehlerberichten für alle am Ende des Sprints nicht behobenen Fehler.

Aufgabenstellung Teil 2: Planung von Sprint 2

Teilaufgabe 2.1: Planungsspiel Sprint 2 (*Sprint Planning Game*) **2 Punkte**

Nachdem Sie die Implementierung (d.h. Codierung, Integration und Testen) für Sprint 1 beendet haben, ist es jetzt an der Zeit, Sprint 2 zu planen. Wie in Teilaufgabe 1.2 erwähnt, haben Sie möglicherweise in Sprint 1 nicht alle Arbeitseinheiten fertigstellen können. Überdies haben Sie möglicherweise Fehler identifiziert. Diese Dinge müssen Sie nun analysieren und auf dieser Grundlage die Sprint-Auftragsliste *(Sprint Backlog)* für Sprint 2 planen. Hierzu führen Sie folgende Tätigkeiten durch:

- Schätzen Sie den Aufwand für die geplanten, aber in Sprint 1 nicht realisierten Arbeitseinheiten neu ab. Überprüfen Sie ebenfalls, aufgrund der in Sprint 1 gemachten Erfahrungen, die Aufwandschätzung für die übrigen noch nicht implementierten Benutzergeschichten und nehmen Sie gegebenenfalls Anpassungen vor.
- Überprüfen Sie die Prioritäten aller noch nicht implementierten Benutzergeschichten und nehmen Sie gegebenenfalls Anpassungen vor (siehe dazu den Hinweis unten).
- Schätzen Sie für jeden Fehlerbericht den Aufwand für die Fehlerbehebung und priorisieren Sie den Fehlerbericht.

- Falls Sie während des ersten Sprints neue Benutzergeschichten entdeckt haben, welche Sie in Sprint 2 realisieren wollen, so schreiben Sie diese Geschichten, schätzen ihren Aufwand ab und priorisieren Sie sie (siehe dazu den Hinweis unten).
- Wählen Sie nun diejenigen Benutzergeschichten und Fehlerberichte aus, welche Sie in Sprint 2 bearbeiten wollen, und erstellen Sie die Auftragsliste (*Sprint Backlog*) für Sprint 2 in tabellarischer Form (gemäss Tabelle 2 aus Übung 3).

Hinweis: In einem realen agilen Projekt sind die Erstellung neuer Benutzergeschichten, die Änderung bestehender, noch nicht implementierter Benutzergeschichten und alle Priorisierungen Aufgabe des Produkteigners. Im Rahmen unseres Übungsprojekts müssen Sie sich dementsprechend in die Rolle des Produkteigners versetzen, wenn Sie Benutzergeschichten bearbeiten oder Aufgaben priorisieren.

Die Lösung dieser Teilaufgabe besteht aus der Tabelle mit der Sprint-Auftragsliste für Sprint 2.

Teilaufgabe 2.2: Aufgabenverteilung für Sprint 2 **2 Punkte**

Wie Sie dies in Übung 3 für Sprint 1 bereits gemacht haben, müssen Sie die neu ausgewählten Benutzergeschichten in einzelne Arbeitseinheiten (*Tasks*) herunterbrechen, welche Sie dann mit Ihren Teammitgliedern untereinander verteilen. Stellen Sie das Ergebnis tabellarisch dar (gemäss Tabelle 3 in Übung 3).

Die Lösung dieser Teilaufgabe besteht aus der Tabelle mit der Aufgabenverteilung (Work Breakdown) für Sprint 2. Die Durchführung von Sprint 2 wird Gegenstand der Übung 5 sein.

Vorbereitung der Sprint Review Sitzung

Die Sprint Review Sitzung am 22.11.2016 dauert exakt 10 Minuten. Sie werden dabei den Produkteigner treffen, welcher beurteilen wird, welche Punkte (Benutzergeschichten, Arbeitseinheiten) als abgeschlossen betrachtet werden können. Die Sitzung ist in drei Teile gegliedert.

1. Sie werden exakt 4 Minuten Zeit haben, um Ihr Produkt zu demonstrieren (falls Sie überziehen, werden Sie unterbrochen/abgewürgt). Nutzen Sie diese Zeit, um Ihr Pilotsystem zu erklären und zu zeigen, welche Benutzergeschichten aus der Sprint-Auftragsliste implementiert sind. Ferner müssen Sie ausgewählte Tests demonstrieren. *Wir empfehlen Ihnen dringend, diese Präsentation gut vorzubereiten und vorher zu üben.*
2. Der zweite Teil beansprucht maximal 3 Minuten. Der Produkteigner wird Ihnen Fragen zu Ihren Entwurfs- und Implementierungsentscheidungen stellen. Diese Fragen werden an einzelne Teammitglieder gerichtet. Sie sollten daher nicht nur den von Ihnen selbst geschriebenen Code kennen, sondern auch über das Gesamtdesign Ihres Pilotsystems Bescheid wissen. Danach haben Sie eine kurze Besprechung zur nächsten Sprint-Planung. Sie werden den Produkteigner bzw. die Produkteignerin einerseits über die Benutzergeschichten informieren, welche Sie in Sprint 1 nicht oder nur teilweise implementieren konnten und nun in Sprint 2 implementieren wollen. Andererseits erläutern Sie die Fehlerberichte aus Sprint 1, welche Sie in Sprint 2 bearbeiten wollen, sowie die neuen Geschichten, welche Sie für Sprint 2 ausgewählt haben.
3. Am Ende der Diskussion werden Sie zusammen mit Ihrem Kursassistenten Ihren Fortschritt beurteilen und diesbezüglich Fragen diskutieren (3 Minuten): was lief gut, was kann künftig besser gemacht werden, was haben Sie gelernt, welche Konsequenzen ziehen Sie? In agilen Projekten nennt man solche Besprechungen “Sprint Retrospective”.

Hinweis: In realen Projekten mischt sich der Produkteigner bzw. die Produkteignerin nicht in die Sprintplanung und die Aufgabenverteilung ein. In der Übungssituation profitieren Sie jedoch mehr, wenn wir Ihnen Feedback zu Ihrer Sprintplanung geben.