HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

[Graphs-Visualization-Service]

Studienarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Autoren
[Michael Wieland]
[Murièle Trentini]

Betreuer
[Thomas Letsch]
Dozent für Informatik an der
Hochschule für Technik Rapperswil

Auditor
[Name]

Zeitraum: 18.09.2017 - 22.12.2017

Abstract

Management Summary

Ausgangslage

Vorgehen, Technologien

Ergebnisse

Ausblick

Technischer Bericht

Ausgangslage & Problembeschreibung

Lösungskonzept

Umsetzung

Ergebnisdiskussion mit Ausblick

Danksagungen

Wir danken folgenden Personen für Ihre Unterstützung während der Studienarbeit:

- Thomas Letsch für die Betreuung unserer Studienarbeit.
- Jessica Martin für die technische Unterstützung beim Logo Design.

Inhaltsverzeichnis

Kapitel 1

[Anforderungsanalyse]

1.1 [Ausgangslage]

1.2 [Mehrwert]

1.3 [Aufgabenstellung]

1.4 [User Stories]

1.5 [Use Cases]

1.5.1 Brief

1.5.2 Fully Dressed

1.6 [Domainanalyse]

Kapitel 2

[Realisierung]

2.1 [Architektur]

2.2 [UI Design]

2.2.1 Logo

Das Logo wurde von den Eigenschaften des Kraken [10] inspiriert. Kraken sind bekannt dafür, dass sie viele Irrgarten-Probleme effizient lösen können. Dies ist eine Anspielung an die Algorithmen, die vom Graphs-Visualization-Service (GVS) unterstützt werden. Ebenfalls wurden die Saugnäpfe des Kraken als Graph visualisiert und auf der Stirn ist ein binärer Baum zu erkennen.

Abbildung 2.1 – Graphs-Visualization-Service Logo

2.2.2 Konzept

2.2.3 Icons

2.2.4 Farben

2.2.5 Wireframes

2.3 [Testing]

Kapitel 3

[Projektmanagement]

3.1 [Zeitplanung]

Das Projekt wird im Rahmen der Studienarbeit durchgeführt. Insgesamt stehen 14 Wochen zur Verfügung in welchen jedes Teammitglied 240 Stunden leisten muss. Somit entstehen ca. 17 Stunden Arbeitsaufwand pro Woche und Teammitglied.

3.2 [Projektverwaltung]

Als Projektmanagement Software wird Jira [9] eingesetzt. In Jira werden alle Requirements als Issues erfasst und in den Product Backlog eingepflegt. Pro Iteration sollen jeweils so viele Issues eingeplant werden, wie unter Berücksichtigung von administrativen Aufgaben abgearbeitet werden können. Dabei spielt der Teamspeed eine grosse Rolle, welcher sich über die Projektdauer einpendeln soll.

3.2.1 Phasen

Das Projekt ist grob in zwei Phasen aufgeteilt.

Analysephase

In der Analysephase wird das bestehende Produkt genau analysiert. Dabei soll festgestellt werden, welche Teile der Software verbessert werden müssen. Ebenfalls wird die Machbarkeit mit kleinen Prototypen validiert.

Realisationsphase

In der Realisationsphase beginnt das eigentliche Software Projekt. Wie dabei vorgegangen wird ist in den folgenden Unterkapitel genauer beschreiben.

3.2.2 Iterationsplanung

Die Iterationsplanung orientiert sich grob am Rational Unified Process (RUP) und ist unterteilt in eine Inception-, Elaboration-, Construction- sowie eine Transition-Phase. Jede Phase besteht aus einer oder mehreren Iterationen, die jeweils 2 Wochen dauern und am Freitag enden. Eine Iteration wird nach SCRUM organisiert. Am Anfang des Sprints definiert das Projektteam die zu erledigen Issues und schätzt deren Aufwand. (Siehe 3.2.4)

Phase	Beschreibung
Inception	Projektsetup (Jira, Workflow, IDE), Einarbeitung in die bestehenden Sourcen
Elaboration 1	Anforderungsspezifikation, Domainmodell
Elaboration 2	Architektur Modell, Guidelines für Quellcode und Tests, Wireframes
Construction 1	
Construction 2	
Construction 3	
Transition	

Tabelle 3.1 – Iterationsplanung

3.2.3 Meilensteine

3.2.4 Schätzungen

Issues werden auf Basis von Story Points geschätzt. Dieses Vorgehen hat sich mit SCRUM etabliert. Die Nutzung von Story Points führt dazu, dass nicht die individuell unterschiedliche Bearbeitungszeit,

sondern die Komplexität einer User Story geschätzt wird. Dies vereinfacht und homogenisiert die Schätzungen und hilft den Teamspeed zu bestimmen.[12, 14]

3.2.5 Zeitauswertung

Für die Zeitauswertung wird das Jira Plugin Tempo [1] verwendet. Dieses bietet umfassende Auswertungsmöglichkeiten, sowie Exports nach MS Excel. Die Auswertungen dieser Arbeit befinden sich im Anhang D.

3.2.6 Meetings

Über die gesamte Projektdauer findet jeweils am Mittwoch um 17:15 ein wöchentliches Standortmeeting statt. Die Beschlüsse aus den Meetings werden protokolliert und bis spätestens 24h später an alle Teilnehmer versendet. Allfälliges Feedback wird nachträglich eingepflegt und versioniert abgelegt.

3.3 [Artefakte]

3.4 [Repositories]

Die Artefakte werden in verschiedenen Repositories auf GitHub versioniert abgelegt. So gibt es für die Dokumentation, die Meetingprotokolle und den Sourcecode jeweils ein eigenes Repository.

3.4.1 Gitflow

Die Branch Struktur orientiert sich an Gitflow [7]. Pro Issue wird ein Feature Branch erstellt, der nach erfolgreichem Review in den Development Branch gemerged wird. Beim Erreichen eines Meilensteins wird der Entwicklungszweig in den Master Branch gemerged.

3.5 [Entwicklungsumgebung]

Als Integrated Development Environment wird Eclipse mit folgenden Plugins verwendet

- FindBugs [6] zur statischen Code Analyse
- EclEmma [5] zur Überprüfung der Test Abdeckung
- Stan4J [11] zur Überprüfung von zyklischen Abhängigkeiten
- Checkstyle [3] zur Überprüfung der Coding Richtlinien.

3.6 [Frameworks]

Als UI Framework wird gemäss Aufgabenstellung JavaFX eingesetzt (siehe Anhang C). JavaFX gilt seit 2014 als Standardlösung für grafische Java Anwendungen. JavaFX ist eine komplette Neuentwicklung und der offizielle Nachfolger vom Abstract Window Toolkit [2] und Swing [13].

3.7 [Continuous Integration]

3.8 [Qualitätsmanagement]

3.8.1 TDD

Zur Sicherung der Software Qualität sind gute Tests unerlässlich. Für jedes Feature werden deshalb zuerst entsprechende Tests geschrieben. Dieses Vorgehen wird Test Driven Development genannt.

3.8.2 Definition of Done

Source Code wird erst zum Review freigegeben, wenn folgende Kriterien erfüllt sind.

- Die IDE zeigt keine Warnungen und Fehler
- Es gibt keinen auskommentierten Code
- Es existieren sinnvolle Unit und Integrationstests für das Feature
- Alle Metriken geben grünes Licht
- Das Issue wurde in Jira [9] zum Review vermerkt

3.8.3 Review

Nach Abschluss eines Issues wird dieses in Form eines Pull-Request an den Teampartner zum Review freigegeben. Durch das Vier-Augen-Prinzip kann die Qualität des Produktes hoch gehalten werden. Zusätzlich haben beide Teampartner stets den selben Wissensstand.

3.8.4 Metriken

3.9 [Risikomanagement]

3.9.1 Backups

Zur Minimierung von allfälligen Datenverlusten wird wie folgt vorgegangen:

1. Täglich automatisiertes Backup aller Projektdaten im JIRA [9] (Zeiterfassung, erstellte Issues, Jira Konfiguration)
2. Die Projektdokumentation sowie der Programmcode wird in den vier Github Repositories der Github Organisation [8] versioniert abgelegt. Für die Projektmitglieder gilt der Grundsatz "Commit early and often".
3. Weitere Artefakte werden auf Dropbox [4] abgelegt und dort automatisch gesichert und versioniert.

Anhang

Glossar

GVS Graphs-Visualization-Service: Titel des vorliegenden Produktes 2, 8

RUP Rational Unified Process: Vorgehensmodell in der Software Entwicklung, dass ursprünglich von IBM entwickelt wurde. 4, 8

SCRUM Vorgehensmodell zur agilen Softwareentwicklung 4, 8

Literatur

[1] *Atlassian Marketplace: Tempo Timesheets for Jira.* URL: https://marketplace.atlassian.com/plugins/is.origo.jira.tempo-plugin/cloud/overview.

[2] *AWT.* URL: https://de.wikipedia.org/wiki/Abstract_Window_Toolkit.

[3] *Checkstyle.* URL: http://checkstyle.sourceforge.net/.

[4] *Dropbox.* URL: https://www.dropbox.com.

[5] *EclEmma.* URL: http://www.eclemma.org/.

[6] *FindBugs.* URL: http://findbugs.sourceforge.net/.

[7] *Gitflow.* URL: http://nvie.com/posts/a-successful-git-branching-model/.

[8] *Github Repositories.* URL: https://github.com/Graphs-Visualization-Service.

[9] *Jira.* URL: https://project.redbackup.org/projects/GVS/.

[10] *Kraken.* URL: https://de.wikipedia.org/wiki/Kraken.

[11] *Stan4J.* URL: http://stan4j.com/.

[12] *Story Points verständlich erklärt.* URL: http://www.ksimons.de/2011/06/story-points-verstandlich-erklart/.

[13] *Swing.* URL: https://de.wikipedia.org/wiki/Swing_(Java).

[14] *The secrets behind story points and agile estimation.* URL: https://www.atlassian.com/agile/estimation.

Abbildungsverzeichnis

Tabellenverzeichnis

Anhang A

Eigenständigkeitserklärung

Der/Die Verfasser/in erklärt hiermit, dass er/sie die vorliegende Arbeit selbständig, ohne fremde Hilfe und ohne Benutzung anderer als die angegebenen Hilfsmittel angefertigt hat. Die aus fremden Quellen (einschliesslich elektronischer Quellen) direkt oder indirekt übernommenen Gedanken sind ausnahmslos als solche kenntlich gemacht. Durch Copyright geschützte Materialien wurden in dieser Arbeit nicht in unerlaubter Weise genutzt. Die Arbeit ist in gleicher oder ähnlicher Form oder auszugsweise im Rahmen einer anderen Prüfung noch nicht vorgelegt worden.

.....	.....
Ort, Datum	Unterschrift des/der Verfassers/in
.....	.....
Ort, Datum	Unterschrift des/der Verfassers/in

Anhang B

Vereinbarung

Mit dieser Vereinbarung werden die Rechte über die Verwendung und die Weiterentwicklung der Ergebnisse der Studienarbeit GVS von Muriele Trentini und Michael Wieland unter der Betreuung von Thomas Letsch geregelt.
Die Urheberrechte stehen der Studentin / dem Student zu.
Die Ergebnisse der Arbeit dürfen sowohl von der Studentin / dem Student wie von der HSR nach Abschluss der Arbeit verwendet und weiter entwickelt werden

.....	.....
Ort, Datum	Unterschrift des/der Verfassers/in
.....	.....
Ort, Datum	Unterschrift des/der Verfassers/in

Anhang C

Aufgabenstellung

Die folgenden Seiten enthalten die offizielle Aufgabenstellung dieser Studienarbeit.

Graphs-Visualization-Service GVS 2.0

Studenten

- Murièle Trentini
- Michael Wieland

Einführung

Bei den Modulen "Algorithmen und Datenstrukturen 1" und "Algorithmen und Datenstrukturen 2" werden Algorithmen zur Bearbeitung von Bäumen und allgemeinen Graphen programmiert.
Dabei ist die Programmierung für die Visualisierung derartiger Algorithmen und den dazugehörigen Datenstrukturen mit relativ grossem Aufwand verbunden und nicht Bestandteil des Modul-Stoffes.

Aus diesem Grund wurde dazu im Jahre 2005 der 'Graphs-Visualization-Service' erstellt. Dieser Service ist für den Einsatz im Unterricht ausgelegt um die Studenten bei der Programmierung zu unterstützen.
Er besteht serverseitige aus einer Java-Applikation und clientseitig steht je ein API für Java und .NET zur Verfügung.

Mit einem Major-Upgrade soll das Tool nun auf die aktuellen Technologien aktualisiert werden.

Aufgabenstellung

Beim aktuellen 'Graphs-Visualization-Service' soll unter Berücksichtigung resp. Beibehaltung der aktuellen Funktionalität:

- serverseitig Swing durch JavaFX ersetzt werden
- clientseitig das API um Generics erweitert werden
- punktuelle Verbesserungen vorgenommen werden

Unter Berücksichtigung von aktuellen Software-Engineering-Methoden soll ein geeigneter Entwicklungsprozess definiert und darauf basierend das Projekt realisiert werden.

Technologien

- Java
- Enterprise Architect

Generelles

- Die Vorgaben der Abteilung Informatik [1] sind einzuhalten, insbesondere die Anleitung zur Dokumentation [2].
- Die "Generelle Richtlinien für Studien- und Bachelorarbeiten" [3] sind einzuhalten.
- Mit dem CASE-Tool Enterprise Architect ist ein UML-Modell zu führen, welches synchron mit den Programm-Sourcen und der Projekt-Dokumentation ist.
- Ein Java-Entwickler muss mit der Projekt-Dokumentation in die Lage versetzt werden, die Applikation in Betrieb zu nehmen und weiter entwickeln zu können.

Termine

- Montag, 18.09.17 Beginn der Studienarbeit
- Freitag, 22.12.17 17:00 Uhr Abgabe der Studienarbeit

Betreuung

- Betreuer
  Thomas Letsch
  tletsch@hsr.ch
  055 - 22 24 567 (HSR Büro 5.204); 055 - 214 43 50 (Geschäft)
- Besprechungen
  Wöchentliche Besprechung jeweils Mittwoch 17:15 Uhr

Referenzen

[1] Skripte-Server: /Informatik/Fachbereich/Studienarbeit_Informatik/SAI
[2] Anleitung Dokumentation BA_SA_170905.pdf
[3] "Generelle Richtlinien für Studien- und Bachelorarbeiten"
(v1.7 / 19.08.2015, Thomas Letsch)

Rapperswil, 18.September 2017

Thomas Letsch

Anhang D

Zeitauswertung

Anhang E

Testspezifikation

Anhang F

Meeting Protokolle