Projektdokumentation

Softwarearchitekt: Philipp Rimmele – philipp.rimmele@stud.htwk-leipzig.de
Product Owner: Anna Heinrich – anna.heinrich@stud.htwk-leipzig.de
Marian Geissler – marian.geissler@stud.htwk-leipzig.de
HTWK Leipzig

Inhaltsverzeichnis

I. Anforderungsspezifikation

I.1 Initiale Kundenvorgaben

Maecenas sed ultricies felis. Sed imperdiet dictum arcu a egestas. In sapien ante, ultricies quis pellentesque ut, fringilla id sem. Proin justo libero, dapibus consequat auctor at, euismod et erat. Sed ut ipsum erat, iaculis vehicula lorem. Cras non dolor id libero blandit ornare. Pellentesque luctus fermentum eros ut posuere. Suspendisse rutrum suscipit massa sit amet molestie. Donec suscipit lacinia diam, eu posuere libero rutrum sed. Nam blandit lorem sit amet dolor vestibulum in lacinia purus varius. Ut tortor massa, rhoncus ut auctor eget, vestibulum ut justo.

I.2 Produktvision

Quisque vel arcu eget sapien euismod tristique rhoncus eu mauris. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Cras ligula lacus, dictum id scelerisque nec, venenatis vitae magna. Cras tristique porta elit, non tincidunt ligula placerat lobortis. Pellentesque quam enim, mattis in cursus eu, blandit et massa. Mauris aliquet turpis blandit elit vehicula sed posuere lectus facilisis. Donec blandit adipiscing tortor, quis lobortis purus eleifend vel. Nam a tellus at magna scelerisque blandit vel nec erat.

I.3 Liste der funktionalen Anforderungen

XXX

I.3.1 Userstories

Vorschau Als Benutzer wünsche ich mir eine Vorschau der Diagramme, damit ich einschätzen kann ob ich damit zufrieden bin.

Interfaces Als Benutzer wünsche ich mir, dass ich abhängig von Interfaces die zugehörigen Klassen anzeigen lassen kann, damit ich weiß, welche Methoden ich implementieren muss.

Kommandozeile Als Benutzer wünsche ich mir, dass das Programm von der Kommandozeile aus aufrufbar ist, um es automatisiert starten zu können.

Dateien einlesen

Art der eingelesenen Datei Als Benutzer wünsche ich mir, dass eine Auswahl zwischen Jar- und Java-Dateien möglich ist, damit Quellcode nicht doppelt eingelesen wird.

Java-Dateien Als Benutzer wünsche ich mir, dass Java-Dateien einlesbar sind, um den Quellcode von einer oder mehreren Klassen zu analysieren.

Jar-Dateien Als Benutzer wünsche ich mir, dass Jar-Dateien einlesbar sind, um den Quellcode zu analysieren.

Installation Als Benutzer wünsche ich mir, dass die Installation unkompliziert ist, damit ich das Programm schnell benutzen kann.

Klassendiagramme Als Benutzer wünsche ich mir, Klassendiagramme aus meinem bestehenden Quellcode erstellen zu können, damit ich das nicht manuell tun muss.

Sequenzdiagramme Als Benutzer wünsche ich mir, Sequenzdiagramme aus meinem bestehenden Quellcode erstellen zu können, damit ich das nicht manuell tun muss.

Klassenauswahl Als Benutzer wünsche ich mir die Möglichkeit, Klassen zu selektieren, damit die Diagramme nicht zu unübersichtlich werden.

Methoden- und Variablenauswahl Als Benutzer wünsche ich mir die Möglichkeit, Methoden und Variablen zu selektieren, damit die Diagramme nicht zu unübersichtlich werden.

Multiple Klassenauswahl Als Benutzer wünsche ich mir einen Button, mit dem ich alle Klassen an- oder abwählen kann, damit ich nicht alle Klassen einzeln auswählen muss.

Multiple Methodenauswahl Als Benutzer wünsche ich mir einen Button, mit dem ich alle Methoden an- oder abwählen kann, damit ich nicht alle Methoden einzeln auswählen muss.

Layout Als Benutzer wünsche ich mir, das Layout meiner Diagramme ändern zu können, um deren Aussehen zu verbessern.

Plugin Als Benutzer wünsche ich mir, PUML als Plugin direkt in Eclipse verwenden zu können, damit ich nicht außerhalb meiner Entwicklungsumgebung arbeiten muss.

Drag-and-Drop Als Benutzer wünsche ich mir, die ausgewählten Dateien oder Ordner per Drag-and-Drop in das Programm aufzunehmen, damit ich den Datei-öffnen-Dialog nicht nutzen muss.

Anzeigen und Speichern von PlantUML Als Benutzer wünsche ich mir, Diagramme als PlantUML-Code anzeigen und speichern zu können, um den Aufbau nachvollziehen zu können.

Speichern von Bilddateien Als Benutzer wünsche ich mir, die erstellten Diagramme als Bilddatei exportieren zu können, um sie in meine Projektdokumentation mit aufzunehmen.

Speichern von Konfigurationen Als Benutzer wünsche ich mir, meine Konfiguration speichern zu können, damit ich meine Präferenz nicht jedes Mal aufs Neue einstellen muss.

Benutzerhandbuch Als Benutzer wünsche ich mir, das Benutzerhandbuch über die GUI anzeigen lassen zu können, damit ich die gedruckte Version nicht benötige.

Übersicht aller Diagramme für ein Projekt Als Benutzer wünsche ich mir eine Übersicht aller Diagramme, die ich für mein Projekt erstellt habe, damit ich leichter auf diese zugreifen kann.

Drucken Als Benutzer wünsche ich mir, Diagramme über das GUI drucken zu können, damit ich die Bilddateien nicht separat öffnen muss.

Exceptions als Sequenzdiagramme Als Benutzer wünsche ich mir, dass der mögliche Pfad der Exceptions als Sequenzdiagramm angezeigt werden kann, um ungehandelte Exceptions zu vermeiden.

Plattformunabhängigkeit Als Project Owner wünsche ich mir, dass das Programm plattformunabhängig ist, damit es sich gut verbreiten lässt.

I.4 Liste der nicht-funktionalen Anforderungen

XXX

I.5 Weitere Zuarbeiten zum Produktvisions-Workshop

XXX

I.6 Zuarbeit von Autor X

XXX

I.7 Zuarbeit von Autor Y

XXX

I.8 Liste der Kundengespräche mit Ergebnissen

Datum	Anliegen oder Fragen	Ergebnisse
02.11.18	Wie genau soll das Layout des Diagramms anpassbar sein?	Das Layout soll sowohl manuell als auch automatisch optimiert werden können.
	Reicht es für den ersten Sprint, wenn PUML als Kommandozeilenprogramm umgesetzt wird?	Es soll möglichst früh eine grafische Oberfläche entwickelt werden. Deren Funktionsumfang darf zu Beginn ruhig minimal sein. Wichtig ist, dass das Team möglichst früh einen „optischen Erfolg“ zu verzeichnen hat.

II. Architektur und Entwurf

II.1 Zuarbeiten der Teammitglieder

XXX

II.2 Entscheidungen des Technologieworkshops

XXX

II.3 Überblick über Architektur

XXX

II.4 Definierte Schnittstellen

XXX

II.5 Liste der Architekturentscheidungen

Zeit	Entscheidung
Bei Projektvergabe	Als Programmiersprache wird Java verwendet. Begründung der Entscheidung: • Platformunabhängigkeit • Alle aus dem Team beherschen Java • Sauber und Einsteigerfreundlich • Weiterentwicklung zu Eclipse-Plugin möglich • Ausgereifte GUI-Frameworks verfügbar
Technologieworkshop	

III. Prozess- und Implementationsvorgaben

III.1 Definition of Done

XXX

III.2 Coding Style

Bitte die Datei javaCodeStyle.xml im specification-Verzeichniss in Eclipse importieren und verwenden. Hierfür in Eclipse unter „Window->Preferences->Java->Code Style->Formatter“ auf Import klicken und die XML-Datei auswählen.
Ist der passende Coding Style eingestellt kann der Quellcode mit „STRG+SHIFT+F“ automatisch formatiert werden. Wird dies vor jedem Commit gemacht, ensteht ein einheitlicher Code-Style und die Änderungen können gut mit GIT überprüft werden.
Des weiteren empfielt es sich bei größeren oder stark geschachtelten Code-Abschnitten die Zuge-

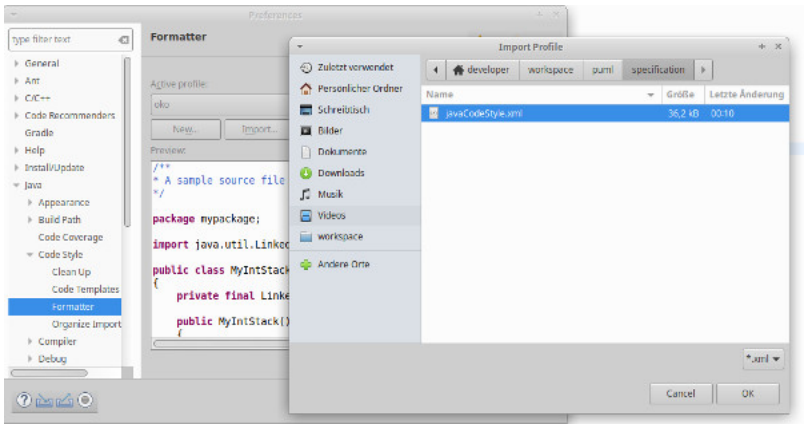

Abbildung 1: Code-Style in Eclipse importieren

hörigkeit der Schließenden Klammer mit einem Kommentar zu Kennzeichnen.
Sonstige Konventionen:

- Variablen und Instanzen beginnen kleingeschrieben
- Klassen und Interfaces beginnen mit Großbuchstaben
- Besteht ein Namen aus mehreren zusammengesetzten Wörtern, beginnen alle weiteren Wörter mit Großbuchstaben (keine Unterstriche in Namen verwenden)
- Aussagekräftige Namen verwenden
- Alle Namen auf Englisch
- Die Kommentare auf Deutsch

III.3 Zu nutzende Werkzeuge

- Eclipse - Entwicklungsumgebung
- GIT - Dateiversionierung
- Meld - Unterschiede zwischen Dateien anzeigen
- Texmaker - Latex-Editor
- GIMP - Bildbearbeitung für das Editieren von Screenshots

IV. Sprint 1

IV.1 Ziel des Sprints

Es soll eine funktionsfähige Basisversion, welche für das einfache erstellen von Klassendiagrammen aus Java-Code verwendet werden soll entstehen. Das Programm soll sowohl über die Kommandozeile, als auch über eine grafische Oberfläche bedient werden können. Die erzeugten Klassendiagramme sollen in der grafischen Oberfläche angezeigt werden können.

IV.2 User-Stories des Sprint-Backlogs

IV.2.1 Dateien einlesen

Art der eingelesenen Datei Als Benutzer wünsche ich mir, dass eine Auswahl zwischen Jar- und Java-Dateien möglich ist, damit Quellcode nicht doppelt eingelesen wird.

Java-Dateien Als Benutzer wünsche ich mir, dass Java-Dateien einlesbar sind, um den Quellcode von einer oder mehreren Klassen zu analysieren.

Jar-Dateien Als Benutzer wünsche ich mir, dass Jar-Dateien einlesbar sind, um den Quellcode zu analysieren.

IV.2.2 Vorschau

Als Benutzer wünsche ich mir eine Vorschau der Diagramme, damit ich einschätzen kann ob ich damit zufrieden bin.

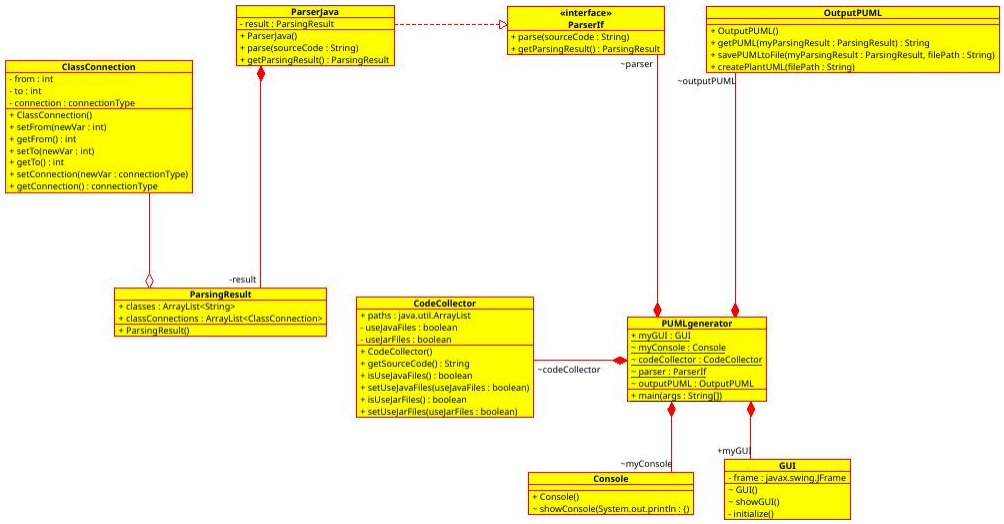

Abbildung 2: Klassendiagramm des Sprints

IV.2.3 Kommandozeile

Als Benutzer wünsche ich mir, dass das Programm von der Kommandozeile aus aufrufbar ist, um es automatisiert starten zu können.

IV.2.4 Klassendiagramme

Als Benutzer wünsche ich mir, Klassendiagramme aus meinem bestehenden Quellcode erstellen zu können, damit ich das nicht manuell tun muss.

IV.2.5 Anzeigen und Speichern von PlantUML

Als Benutzer wünsche ich mir, Diagramme als PlantUML-Code anzeigen und speichern zu können, um den Aufbau nachvollziehen zu können.

IV.2.6 Plattformunabhängigkeit

Als Project Owner wünsche ich mir, dass das Programm plattformunabhängig ist, damit es sich gut verbreiten lässt.

IV.3 Liste der durchgeführten Meetings

- Planning-Meeting

IV.4 Ergebnisse des Planning-Meetings

Dem gesammten Team ist die geplante Grundstruktur des Programms bekannt. Jeder weis welchen Teil des Programms er implementieren soll.

IV.5 Aufgewendete Arbeitszeit pro Person+Arbeitspaket

Arbeitspaket	Person	Start	Ende	h	Artefakt
Dummyklassen	Musterstudi	3.5.09	12.5.09	14	Klasse.java
AP XYZ					

IV.6 Konkrete Code-Qualität im Sprint

XXX

IV.7 Konkrete Test-Überdeckung im Sprint

XXX

IV.8 Ergebnisse des Reviews

XXX

IV.9 Ergebnisse der Retrospektive

XXX

IV.10 Abschließende Einschätzung des Product-Owners

XXX

IV.11 Abschließende Einschätzung des Software-Architekten

XXX

IV.12 Abschließende Einschätzung des Team-Managers

XXX

V. Sprint 2

VI. Dokumentation

VI.1 Handbuch

XXX

VI.2 Installationsanleitung

XXX

VI.3 Software-Lizenz

XXX

VII. Projektabschluss

VII.1 Protokoll der Abnahme und Inbetriebnahme beim Kunden

XXX

VII.2 Präsentation auf der Messe

Poster, Bericht

VII.3 Abschließende Einschätzung durch Product-Owner

XXX

VII.4 Abschließende Einschätzung durch Software-Architekt

XXX

VII.5 Abschließende Einschätzung durch Team-Manager

XXX