TH Brandenburg
Online Studiengang Medieninformatik
Fachbereich Informatik und Medien
Softwaretechnik
Prof. Dr-Ing. Martin Schafföner

Einsendeaufgabe 1: Requirements Engineering

Sommersemester 2021
Abgabetermin 25.04.2021

Maximilian Schulke
Matrikel-Nr. 20215853

Inhaltsverzeichnis

1 Aufgabenstellung

Folgende Aufgabenstellung wurde im Moodle-Kurs bekannt gegeben:

> Wählen Sie sich ein kleines Entwicklungsprojekt für das gesamte Studienmodul Softwaretechnik und verwenden Sie dieses Projekt für alle Lerneinheiten. Beispielsweise ein Tool zur Erfassung von Requirements oder ein anderes Tool, das Ihnen im Bereich der Softwaretechnik helfen kann, oder ein Tool, welches Sie schon immer mal bauen wollten. Im Verlauf des Moduls können Sie alle Themen wie beispielsweise Modellierung oder Testen anhand dieses Projekts praktisch erproben.
>
> Ermitteln und dokumentieren Sie die Requirements für ihr Projekt!
>
> - Formulieren Sie die Anforderungen als User Stories in der Sprache des Benutzers!
> - Formulieren Sie Abnahmeszenarien!
> - Reichern Sie weitere Informationen an, um die Qualitätskriterien für gute Anforderungen zu erfüllen!
>
> Stellen Sie weiterhin dar, wie, d.h. mit welchen Werkzeugen bzw. in welcher Struktur, Sie diese Requirements verwalten!
>
> Die Abgabe erfolgt in Form genau einer PDF-Datei. Falls sie Werkzeuge einsetzen, die keinen PDF-Export erlauben, fertigen Sie Screenshots an, die Sie mit der übrigen Beschreibung in eine PDF-Datei einfügen.
>
> Anmerkungen:
>
> - Der zeitliche Umfang dieser Einsendeaufgabe wird mit durchschnittlich 4 Stunden abgeschätzt
> - Eine Nachbearbeitung der abgegeben Lösung nach der Deadline ist nicht vorgesehen. Falls Sie Fragen zur Aufgabenstellung haben, fragen Sie diese also bitte im Vorfeld!

2 Projekt

Als Projekt für dieses Semester verwende ich eine Idee, die mir schon seit längerem im Kopf rumgeistert – einen kleinen, minimalistischen *Tiling Window Manager* für Linux bzw. X11, der deutlich flexibler und entwicklerfreundlicher ist als bestehende, vergleichbare Software.

2.1 Zusammenfassung der Domain

Für den Fall, dass Sie noch keinen Kontakt mit dieser Domain hatten, gehe ich im folgenden kurz darauf ein, welche Aufgaben ein *Tiling Window Manager* (kurz. *TWM* oder nur *WM*) typischer Weise übernimmt. Die generelle Aufgabe eines WM's besteht hauptsächlich darin sich um die Kommunikation mit dem *Window-Server* (i.d.R. X11 oder Wayland auf Linux, Quartz Compositor auf MacOS) zu kümmern, und die tatsächliche Anordnung der Fenster auf dem Bildschirm zu regeln (in Ebenen, Kacheln etc. – die Möglichkeiten sind nahe zu unbegrenzt). Typische *Window Manager* von z.B. MacOS können diverse, für uns als Endnutzer mittlerweile als üblich angesehene, Anordnungen wie Floating, Split-Screen und Fullscreen realisieren.

Ein *Tiling Window Managers* hat nun die Besonderheit, dass er anders als von Windows, MacOS oder diversen Linux Desktop-Umgebungen bekannt, die Aufteilung der Fenster automatisch und bestmöglich regelt. Typischerweise hat ein *TWM* eine feste Konfiguration mit Layouts,

in die er die Fenster einsortieren kann (z.B. “Master and Stack“). Somit muss sich der Anwender eines solchen *TWM's* (zumindest initial) nicht selber um die Anordnung seiner Fenster kümmern. Desweiteren zählt die Möglichkeit, Keybindings zur Navigation oder Veränderung der Aufteilung zu definieren, zu den typischen Features eines *Tiling Window Managers*.

2.2 Projekt-Ziel

Das Problem von bestehenden *Window Managern* ist in der Regel deren Alter (bspw. ist das Projekt “Toms Window Manager“ im Jahr 1987 entstanden). Aufgrund des Alters sind viele dieser *Window Manager* noch in C, C++ oder einer vergleichbaren Sprache verfasst worden und mit der Zeit immer weiter gewachsen. Dies macht es deutlich schwerer als es sein müsste, grundlegende Änderungen, die nicht von den Entwicklern vorhergesehen waren, vorzunehmen.

Ziel ist es einen *minimalen Window Manager* zu entwickeln, der *durch eigenen Programm-Code konfiguriert* und erweitert werden kann. Dieses Konzept ist schon öfter Implementiert worden (als Inspiration für dieses Modell dienen die Projekte *XMonad*[1] und *DWM*[2]), allerdings sind diese Projekte leider meistens schlecht dokumentiert und haben i.d.R. unnötig komplexe *API's*. Zu den typischen Anwendern dieser *Window Manager* gehören Entwickler und Power User.

3 Erfassung der Anforderungen

Zur Erfassung der Anforderungen verwende ich das Projektmanagement Feature von GitHub. Dieses hat zumindest im kleinen Rahmen diverse Vorteile gegenüber herkömmlicher Software für Agile Projekte (z.B. Jira von Atlassian). Unter anderem Folgende:

- Übersichtlicher durch kleineren Umfang und somit leichter zu Bedienen
- Direkte Integration mit dem Entwicklungsprozess (Issues können z.B. geschlossen werden sobald ein Feature gemerged wurde)
- Nutzer der Software können direkt Issues anlegen

Diese Entscheidung hat keine Endgültigkeit. Wenn es ein echtes Team mit einem klaren Vorgehensmodell gibt, können andere Tools besser geeignet sein. Außerdem sei erwähnt, dass Jira mit diversen Integrationen erweitert werden kann, um die oben beschriebene Funktionalität nachzurüsten. Zur geordneten Erfassung von Anforderungen, deren Priorisierung, Kategorisierung und Abnahme, reicht *GitHub Projects* allerdings vorerst vollkommen aus. Es gibt die Möglichkeit Sprint oder Kanban Boards anzulegen, diese zu automatisieren oder komplett eigene Prozesse zu integrieren.

4 Anforderungen

[1] wurde 2007 erstmals veröffentlicht. Wurde in Haskell geschrieben und wird auch damit konfiguriert. Siehe `https://xmonad.org/` für weitere Informationen.

[2] ist ein äußerst kleiner (unter 2000 Zeilen Source-Code) Window Manager geschrieben in C. Er hat keine Konfigurationsdatei und wird durch Patchen des Codes konfiguriert. Siehe `https://dwm.suckless.org/` für weitere Informationen.

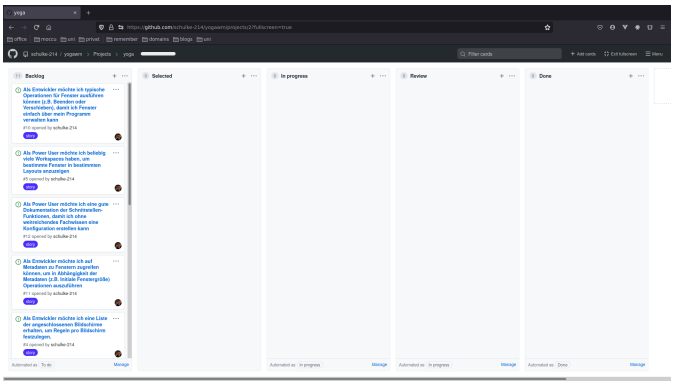

Abbildung 1: *GitHub Projects* Board

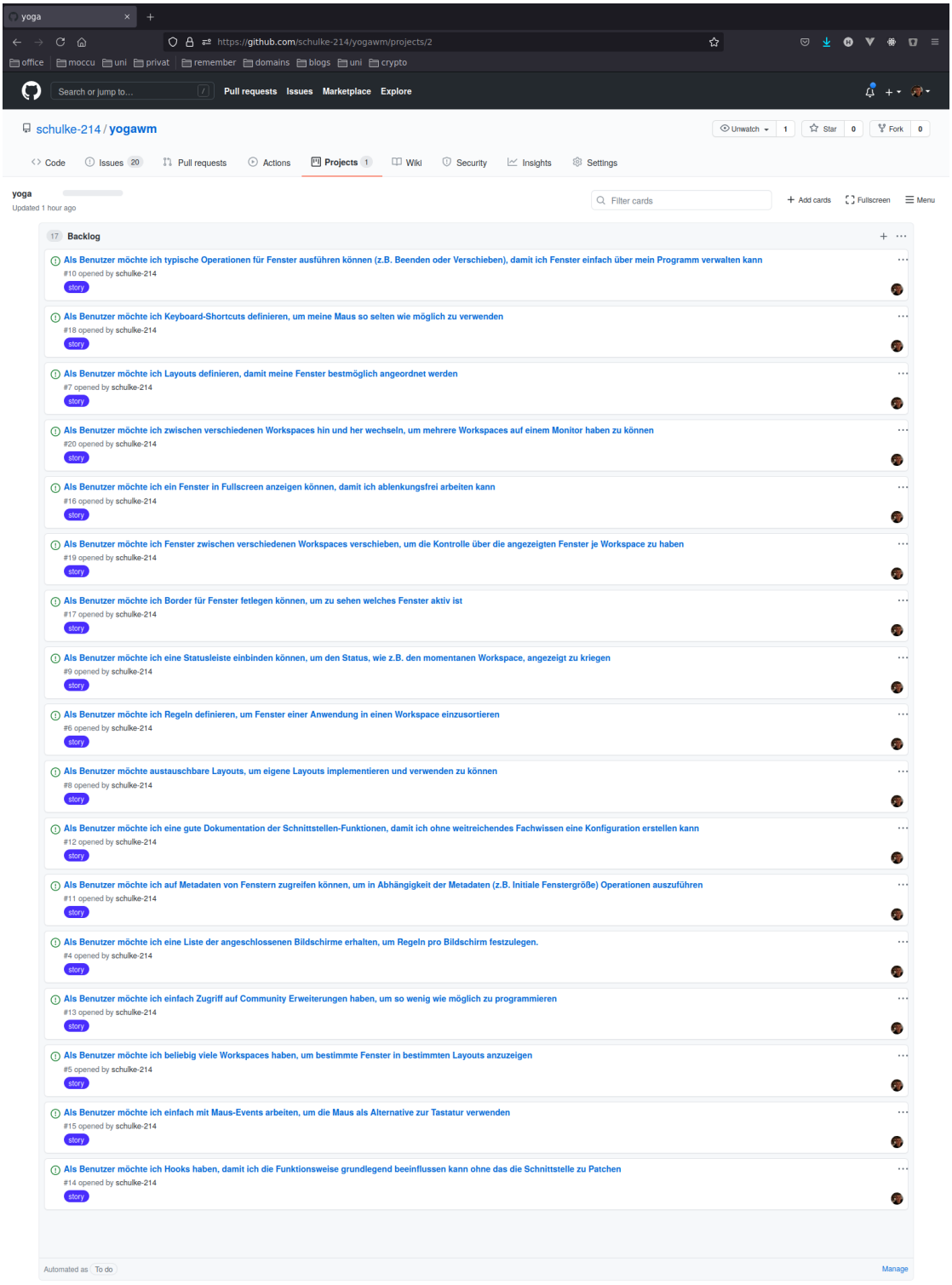

Abbildung 2: *Product Backlog*