Konzepte zur Bachelorarbeit

Template-basierte Synthese von
Verzweigungsstrukturen mittels L-Systemen

Adrian Helberg

1. November 2020

Inhaltsverzeichnis

1 Softwareprojekt

1.1 Vorgehensmodell

Eine Fallstudie der Universität Karlsruhe[5] untersucht den Einsatz der Softwaretechnik **Extreme Programming** (XP) im Kontext der Erstellung von Abschlussarbeiten im Universitätsumfeld.

Hierzu werden folgende Schlüsselpraktiken untersucht:

- XP als Softwaretechnik zur schrittweisen Annäherung an die Anforderungen eines Systems
- Änderung der Anforderungen an das Systems
- Funktionalitäten (**Features**) werden als Tätigkeiten des Benutzers (**User Stories**) definiert
- Zuerst werden Komponententests (Modultests) geschrieben und anschließend die Features (Test-driven Design)
- Keine seperaten Testing-Phasen
- Keine formalen Reviews oder Inspektionen
- Regelmäßige Integration von Änderungen
- Gemeinsame Implementierung (Pair Programming) in Zweiergruppen

Aus der Fallstudie geht hervor, dass Extreme Programming einige Vorteile bei der Bearbeitung eines Softwareprojektes einer Bachelorarbeit bietet. Zum einen können sich Anforderungen an das zu erstellende System durch parallele Literaturrecherche ändern, zum anderen können die Arbeitspakete durch Releases abgedeckt werden.

1.2 Vorgehen

Das Programm zu dieser Arbeit wird mit einem XP-basierten Ansatz erarbeitet. Hierbei beinhaltet ein **Release** Funktionen, die insgesamt für eine neue Version des Systems ausreichen; also ein vollständig funktionsfähiges Programm liefern. **User Stories** sind innerhalb der Iterationen umzusetzende Teilaufgaben und deren Aufwandseinschätzung gibt Auskunft über den Entwicklungsaufwand einer Umsetzung.

Umsetzung des Softwareprojektes in Iterationen mit folgenden Phasen:

- Planung:
  - Release-Planung:
    „*Welche Features werden in diesem Release umgesetzt?*“,
    User Stories, Aufwandsschätzung, Anforderungsmanagement
  - Iterationsplanung:
    Umwandlung der User Stories in kleine Arbeitsschritte,
    Festlegen der Dauer einer Implementierung
- Entwurf: Architektur, Klassendiagramme, Schnittstellen
- Testing: (Automatisierte) Modul- und Regressionstests
- Programmierung: Umsetzung der Features, Implementierung, Modularisierung

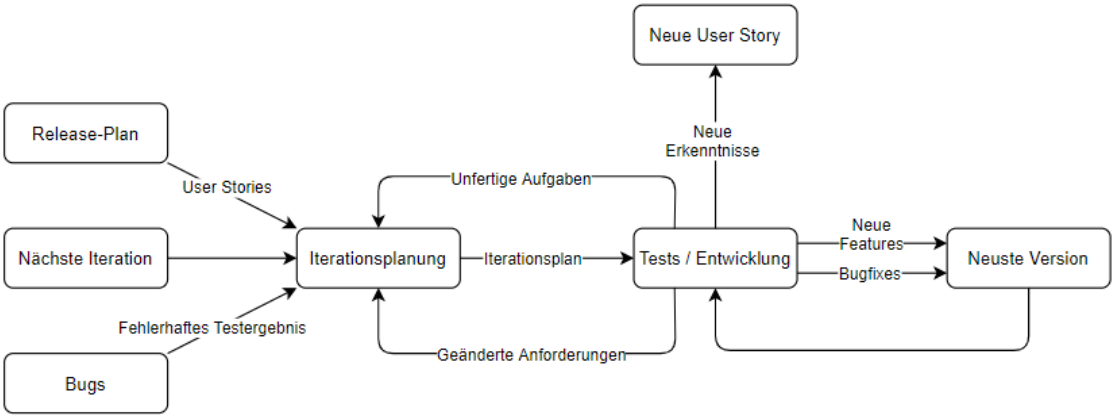

Abbildung 1: Ablaufdiagramm

1.3 Technnologien

- Programmiersprache: Java Version X mit
  - JavaFX Version X
- Build-Management-Tool: Gradle[3] Version X
- Versionskontrolle: Github Repository[2] via Git[1]
- IDE: JetBrains IntelliJ IDEA[4] 2020.2.2 (Ultimate Edition)
- Betriebssystem: Microsoft Windows 10 Pro 64 Bit
- Prozessor: Intel Core i5-3570K CPU @ 3.40GHz
- User Story Map: Trello Board[7]

2 Dokumentation

2.1 Gliederung

Im Folgenden wird eine vorläufige Gliederung der schriftlichen Ausarbeitung gezeigt

1. Abbildungs- und Tabellenverzeichnis
2. Abkürzungsverzeichnis
3. Einleitung
   - 3.1. Problemstellung
   - 3.2. Ziele
   - 3.3. Methodik
   - 3.4. Aufbau
4. Grundlagen
   - 4.1. Grundbegriffe
   - 4.2. Grundlegende Arbeiten
   - 4.3. Verwandte Arbeiten
5. Konzepte
   - 5.1. Probleme & Lösungsansätze
   - 5.2. Architektur
   - 5.3. Algorithmen
6. Implementierung
7. Evaluierung
   - 7.1. Testumgebung
   - 7.2. Beobachtungen & Ergebnisse
   - 7.3. Diskussion und Bewertung
8. Ausblick
9. Literaturverzeichnis
10. Eidesstattliche Erklärung

3 Releases

Dieses Kapitel beschreibt die Release-Planung des Softwareprojekts im Sinne eines XP-orientierten Ansatzes:
Zuerst werden die Funktionalitäten, die im entsprechenden Release umgesetzt werden sollen definiert und **Epics** zugeteilt. Anschließend werden feingranulare User Stories formuliert, die mit einer Aufwandseinschätzung versehen werden. Parallel findet das Anforderungsmanagement (*engl. Requirements Engineering*), das sich mit der Steuerung, Kontrolle und Verwaltung der Anforderungen an das System beschäftigt, statt. Mit diesen Informationen lässt sich dann die Iterationsplanung umsetzen. Die Umwandlung der User Stories in Arbeitsschritte (**Tasks**) und das Festlegen deren Dauer wird hier umgesetzt.

3.1 Release 1

Erstellung einer grafischen Benutzeroberfläche zur Erstellung von Verzweigungsstrukturen
Siehe Exposé Kapitel 1.1.2 Überblick Punkt I. Strukturieren und II. Visualisieren

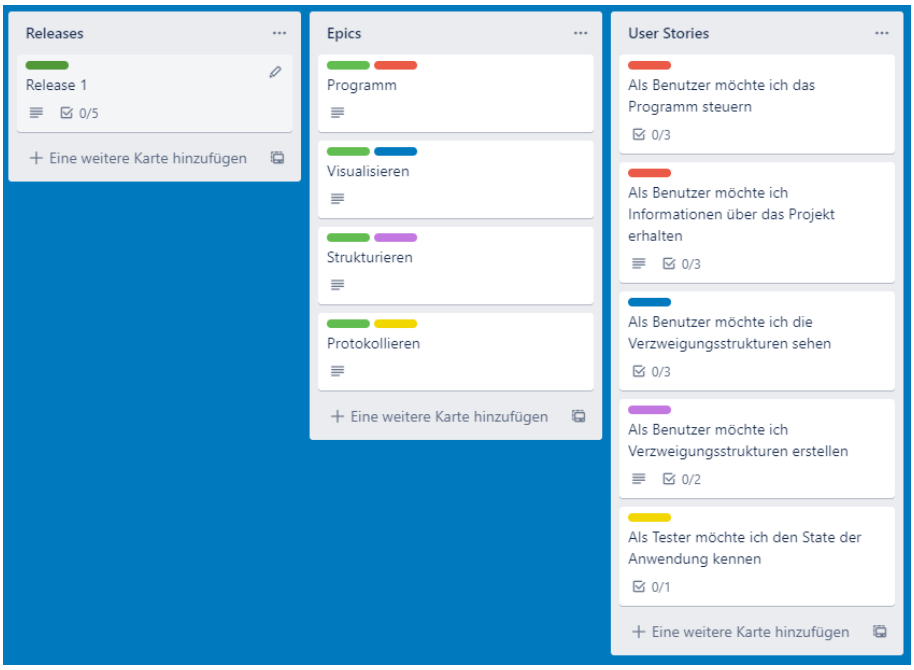

Abbildung 2: User Story Map mit Releases, Epics und User Stories

3.2 Release 2

4 Implementation

Da die Arbeitspakete (Releases) iterativ und abgeschlossen erarbeitet werden, wird das **Pipeline** Design Pattern[6] genutzt.

Quellen

[1] Git - the stupid content tracker. `https://git-scm.com/`. Git is a member of Software Freedom Conservancy.

[2] Github. `https://github.com/adrian-helberg/bachelor`. GitHub, Inc. (2020).

[3] Gradle build tool. `https://gradle.org/`. Gradle Inc. 2020.

[4] Jetbrains intellij idea. `https://www.jetbrains.com/idea/`. Capable and Ergonomic IDE for JVM.

[5] M. M. Muller and W. F. Tichy. Case study: extreme programming in a university environment. In *Proceedings of the 23rd International Conference on Software Engineering. ICSE 2001*, pages 537–544, 2001.

[6] Pipeline design pattern. `https://java-design-patterns.com/patterns/pipeline/`. Datenverarbeitung in mehreren sequenziellen Schritten.

[7] Trello board. `https://trello.com/`. Trello Board, 2020 Atlassian.