Scrum-Dokumente

Version 1.1.0

Gruppe 5 (Patrick Bucher, Pascal Kiser, Fabian Meyer, Sascha Sägesser)

07.04.2018

Inhaltsverzeichnis

1 Stories (nach Epics)

1.1 Epic Log-Ereignisse

1. Aufzeichnung Spielverlaufs: Als *Anwender* möchte ich *persistent festgehaltene Spielereignisse*, um *den Spielverlauf bei Bedarf nachvollziehen zu können*.
2. Aufzeichnung von Problemen: Als *Anwender* möchte ich *persistent festgehaltene Fehlermeldungen*, um *auftretende Probleme nachweisen zu können*.
3. Aufzeichnung von Exceptions: Als *Entwickler* möchte ich *persistent festgehaltene StackTraces*, um *geworfene Exceptions nachvollziehen zu können*.

1.2 Epic Log-Level

4. Message-Level: Als *Entwickler* möchte ich *zu jeder Log-Meldung ein Message-Level setzen können*, um *Log-Meldungen auf verschiedenen Levels definieren zu können*.
5. Codiertes Filter-Level: Als *Entwickler* möchte ich *zur Laufzeit einen Filter-Level setzen können*, um *die Ausgabe der Log-Meldungen steuern zu können*.
6. Konfiguriertes Filter-Level: Als *Administrator* möchte ich *vor der Ausführung einen Filter-Level konfigurieren können*, um *die Ausgabe der Log-Meldungen steuern zu können*.

1.3 Epic Log-Interface

7. Logger-Interface: Als *Entwickler* möchte ich *ein definiertes Logger-Interface zur Verfügung haben*, um *eine Logger-Komponente umsetzen zu können*.
8. Logger: Als *Entwickler* möchte ich *einen* `Logger` *zur Verfügung haben*, um *Log-Meldungen aus dem Game ausgeben zu können*.
9. Logger-Setup: Als *Entwickler* möchte ich *ein* `LoggerSetup` *zur Verfügung haben*, um *Zugriff auf eine* `Logger`*-Instanz zu bekommen*.

1.4 Epic Architektur

10. Komponenten-Austausch: Als *Administrator* möchte ich *eine austauschbare und plattformunabhängige Logger-Komponente*, um *die Logger-Komponente zur Laufzeit ohne Code-Anpassungen*

und Neukompilation austauschen zu können.

11. StringPersistor: Als *Entwickler* möchte ich *eine* `StringPersistor`*-Implementierung für Textdateien (*`StringPersistorFile`*) zur Verfügung haben*, um *geloggte Meldungen persistent in Textdateien festhalten zu können*.
12. Mehrere Log-Clients: Als *Administrator* möchte ich *einen Logger-Server, auf den mehrere Logger-Komponenten parallel loggen können*, um *einen Server für mehrere Clients zur Verfügung stellen zu können*.
13. Payload-Adapter: Als *Entwickler* möchte ich *einen Adapter für die Übergabe des Payloads*, um *Daten in strukturierter Form dem* `StringPersistor` *übergeben zu können*.
14. Adapter-Tests: Als *Entwickler* möchte ich *Unittests für den Adapter haben*, um *zu sehen, ob Änderungen am Adapter zu Problemen führen*.
15. Speicherformat-Strategien: Als *Entwickler* möchte ich *über leicht austauschbare Log-Strategien verfügen*, um *das Speicherformat für Textdateien einfach auf Code-Ebene wählen zu können*.

1.5 Epic Persistenz

16. Log-Aufzeichnung: Als *Administrator* möchte ich *eine verlässliche Aufzeichnung der Log-Ereignisse auf dem Server*, um *die Ereignisse aus dem Spiel in korrekter Reihenfolge nachverfolgen zu können*.
17. Unterscheidung Log-Clients: Als *Administrator* möchte ich *nach Client-Instanz unterscheidbare Logdateien*, um *die Log-Meldungen verschiedener Clients auseinanderhalten zu können*.
18. Textdatei-Log: Als *Administrator* möchte ich *Log-Ereignisse in einfachen Textdateien festhalten*, um *diese mit gängigen Werkzeugen betrachten und auswerten zu können (*`grep`*,* `tail`*, etc.)*.

1.6 Epic Format

19. Speicherung Log-Quelle: Als *Administrator* möchte ich *die Quelle einer Logmeldung sehen*, um *verschiedene Clients und Sessions voneinander unterscheiden zu können*.
20. Speicherung Log-Level: Als *Administrator* möchte ich *den Level einer Logmeldung sehen*, um *diese je nach Bedarf nachträglich filtern zu können*.
21. Speicherung Ereignis-Zeitstempel: Als *Administrator* möchte ich *den Zeitstempel der Message-Erstellung* sehen, um *den genauen Zeitpunkt des Ereignisses zu kennen*.
22. Speicherung Log-Zeitstempel: Als *Administrator* möchte ich *den Zeitstempel des Message-Eingangs auf dem Server* sehen, um *Verzögerungen beim Logging-Vorgang erkennen zu können*.
23. Speicherung Message-Text: Als *Administrator* möchte ich *den Text der Log-Message auf dem Server sehen*, um *das aufgetretene Ereignis erkennen zu können*.

1.7 Epic Konnektivität

24. Konfiguration Serverzugriff: Als *Administrator* möchte ich *die Erreichbarkeit des Servers mittels Konfigurationsdatei definieren können*, um *die Verbindung zum Log-Server ohne Quellcodeanpassung definieren zu können*.

25. Robuste Netzwerkanwendung: Als *Anwender* möchte ich *eine Anwendung mit robuster Netzwerkkonnektivität*, um *auch bei Netzwerkunterbrüchen einen ununterbrochenen Spielfluss ohne verlorene Log-Meldungen zu haben*.
26. Message-Queue: Als *Entwickler* möchte ich *auftretende Log-Ereignisse in einer Message-Queue zwischenspeichern können*, um *Log-Ereignisse bei einem Verbindungsunterbruch erneut senden zu können*.

1.8 Epic Viewer

27. Viewer: Als *Anwender* möchte ich *einen Viewer für Logmeldungen haben*, um *Logmeldungen ohne Kenntnis des physischen Speicherorts auf dem Server betrachten zu können*.

1.9 Rollen

- Ein *Entwickler* hat Zugriff auf den Quellcode und verfügt über die Fähigkeiten und Berechtigungen um Änderungen daran vorzunehmen.
- Ein *Administrator* hat Zugang auf beide konzeptionell involvierten Systeme (Client und Server) und Zugriff auf die Konfigurationsdateien aller involvierter Komponenten.
- Ein *Anwender* kann das Spiel auf dem Client ausführen, kann aber keine Änderungen an Quellcode und Konfigurationsdateien vornehmen und nicht auf den Server zugreifen.

2 Definition of Done

- Funktionalität vom Product Owner abgenommen.
- Code-Review von mindestens einem Teammitglied durchgeführt.
- Bestehende Tests erfolgreich durchgelaufen.
- Eingeführte Features sind in der Projektdokumentation vermerkt.
- Änderungen in GitLab eingecheckt.
- Build auf Jenkins funktioniert.

3 Sprintreviews

3.1 Sprintreview - Sprint 1

3.1.1 Protokoll Sprintreview

Nr. / ID	Titel / Kurzbeschreibung	Version	Status
1	Projektmanagementplan	1.0.0	erledigt
2	Projektstrukturplan	1.0.0	erledigt

Nr. / ID	Titel / Kurzbeschreibung	Version	Status
3	Rahmenplan	1.0.0	erledigt
4	Risikoanalyse	1.0.0	erledigt
5	Scrum-Stories	1.0.0	erledigt

Im ersten Sprint konnten wir nicht alle Stories, welche ursprünglich für den ersten Sprint geplant waren, umsetzen.

Dies liegt daran, dass wir sehr viel Zeit für die Einarbeitung in den Stoff, die Gruppenorganisation und -kommunikation gebraucht haben. Weiterhin benötigte auch das Aufsetzen der Umgebung mitsamt allen Tools eine gewisse Zeit.

Im ersten Sprint konnten wir also folgende Artefakte bereitstellen: Projektmanagementplan, Projektstrukturplan, Rahmenplan, Risikoanalyse, Scrum-Stories.

3.1.2 Testprotokoll

Im Sprint 1 wurden noch keine Tests vorgenommen.

3.2 Sprintreview - Sprint 2

3.2.1 Protokoll Sprintreview

Nr. / ID	Dokument	Version	Status
1	Testplan	1.0.0	erledigt
2	SchnittstelleTCP	1.0.0	erledigt

Nr. / ID	Scrum-Story (ScrumDo)	Status
V6-10	Komponentenaustausch	Doing
V6-28	Logger-Aufrufe zum Festhalten des Spielverlaufs	Done
V6-29	Logger-Aufrufe für Aufzeichnung von Problemen	Done
V6-01	Aufzeichnung des Spielverlaufs	Done
V6-04	Message-Level	Done
V6-17	Unterscheidung Log-Clients	Done
V6-13	Payload-Adapter	Done
V6-20	Speicherung Log-Level	Done
V6-22	Speicherung Log-Zeitstempel	Done
V6-14	Adapter-Tests	Done
V6-21	Speicherung Ereignis-Zeitstempel	Done
V6-08	Logger	Done

Nr. / ID	Scrum-Story (ScrumDo)	Status
V6-11	StringPersistor	Done
V6-02	Aufzeichnung von Problemen	Done
V6-09	Logger-Setup	Done
V6-18	Textdatei-Log	Done
V6-30	TCP-Netzwerk-Interface	Done
V6-24	Konfiguration Serverzugriff	Done
V6-23	Speicherung Message-Text	Done
V6-03	Aufzeichnung von Exceptions	Done
V6-05	Codiertes Filter-Level	Done
V6-07	Logger-Interface	Done
V6-06	Konfiguriertes Filter-Level	Done

Im zweiten Sprint konnten sämtliche Scrum-Stories umgesetzt werden, welche für diesen Sprint vorgesehen waren. Zusätzlich konnte der Rückstand aus Sprint 1 aufgeholt werden. Als Artefakte konnten der Testplan und die TCP-Schnittstelle dokumentiert werden.

Im zweiten Sprint konnten wir also folgende Artefakte bereitstellen: Testplan und SchnittstelleTCP. Weiterhin konnten von 23 Scrum-Stories 22 fertiggestellt werden. Der Komponentenaustausch steht noch aus.

3.2.2 Testprotokoll

3.2.3 Protokoll Review - Sprint 3

3.2.4 Protokoll Review - Sprint 4

3.2.5 Testprotokoll

Im Sprint 1 wurden noch keine Tests vorgenommen.

4 Meilensteinberichte

4.1 Meilensteinbericht 1

4.1.1 Zeitpunkt Meilenstein 1: Beginn SW04

4.1.2 Beschreibung Meilenstein 1

Der erste Meilenstein wurde erreicht, nachdem das Projekt bekannt gegeben wurde, die Gruppen gebildet wurden, alle nötigen Informationen abgegeben wurden, die Gruppen sich mit der Aufga-

benstellung auseinandergesetzt haben und die Gruppen die nötige Entwicklungsumgebung und alle wichtigen Werkzeuge aufgesetzt haben.

Leider haben die Arbeiten des Meilenstein 1 ein wenig mehr Zeit als geplant in Anspruch genommen.

4.1.3 Detail-Vorgaben zum Meilenstein 1

4.1.3.1 Projektmanagement/Projektorganisation

- Die Grundstruktur des Projektmanagementplans anhand der Vorlage der HSLU steht
- Der Organisationsplan steht
- Die Rollen und Zuständigkeiten sind definiert

4.1.3.2 Projektführung

- Der Projektstrukturplan steht
- Der Rahmenplan steht
- Der Iterationsplan steht
- Der Meilensteinplan steht
- Der Ressourcenplan steht
- Mind. ein Vorgehen zur Projektkontrolle wird eingesetzt
- Der ProductBacklog für Meilenstein 1 steht
- Der SprintBacklog für Sprint 1 und 2 steht
- Das Risikomanagement steht

4.1.3.3 Projektunterstützung

- Tools für Entwicklung, Test & Abnahme sind definiert

4.1.3.4 Testplan

- Erste Gedanken zum Testdesign gemacht
- Erste Gedanken zu den Testfällen gemacht

4.1.3.5 Anhänge

- Meilensteinreview für Meilenstein 1 verfasst

4.1.3.6 Systemspezifikation

- Die Grundstruktur des Dokuments steht##### Applikation
- Alle Teammitglieder haben ihre IDE für den Projektstart eingerichtet
- Interface-Definition wurde durch das Interface-Komitee erstellt

4.1.3.7 Meilensteinziele:

1. Projektauftrag als Gruppe entgegennehmen
2. Gruppe setzt sich mit Projektauftrag auseinander
3. Jedes Gruppenmitglied setzt die Entwicklungsumgebung und alle nötigen Werkzeuge auf
4. Alle Vorgaben des Meilensteins 1 werden erfüllt (siehe oben)

4.1.3.8 Wurden die Meilensteinziele erreicht?

1. Ja. Die Gruppenmitglieder waren alle jederzeit anwesend und konnten den Projektauftrag erfolgreich entgegennehmen.
2. Ja. Die Gruppe hat sich kennengelernt und erste Aufgaben wurden verteilt. Jedes Teammitglied hat sich selbständig mit der Thematik auseinandergesetzt.
3. Ja. Jedes Teammitglied hat die Entwicklungsumgebung und die Tools aufgesetzt.
4. Ja. Sämtliche Vorgaben wurden erfüllt.

4.2 Meilensteinbericht 2

4.2.1 Zeitpunkt Meilenstein 2: Beginn SW08

4.2.2 Beschreibung Meilenstein 2

Der zweite Meilenstein wurde erreicht, nachdem das Logger-Interface zur Verfügung gestellt wurde, die Logger- und Stringpersistor-Komponenten implementiert wurden, der Logger-Server implementiert wurde und so die Aufgabe durchgeführt werden konnte, bestimmte Ereignisse der Applikation (Game of Life) mithilfe der selbstgeschriebenen Logger-Klasse zu loggen und die Logs mithilfe der Stringpersistor-Klasse zu persistieren.

4.2.3 Detail-Vorgaben zum Meilenstein 1

4.2.3.1 Projektmanagement/Projektführung

- Der Ressourcenplan ist aktuell
- Die Projektkontrolle ist aktuell
- Der ProductBacklog für Meilenstein 2 steht
- Der SprintBacklog für Sprint 3 und 4 steht

4.2.3.2 Projektunterstützung

- Tools für Entwicklung, Test & Abnahme sind aktuell
- Konfigurationsmanagement ist aktuell
- Releasemanagement ist aktuell

4.2.3.3 Testplan

- Testdesign ist vollständig definiert
- Testfälle für implementierter Code sind vorhanden

4.2.3.4 Anhänge

- Meilensteinreview für Meilenstein 2 verfasst
- Sprintreview für Sprint 1 verfasst
- Sprintreview für Sprint 2 verfasst

4.2.4 Systemspezifikation

- TODO (weitere Infos folgen in späteren Wochen)

4.2.5 Applikation

- Der Logger-Komponente ist als Komponente mithilfe von Interfaces austauschbar
- Die Applikation (Game of Life) gibt bei jedem Log einen Message-Level mit
- Die API der Logger-Komponente kann einen Level-Filter setzen, um die zu übertragenden Logs einzuschränken
- Der Level-Filter kann während der Laufzeit geändert werden
- Die Logs werden durch Logger-Komponenten und den Logger-Server kausal und verlässlich aufgezeichnet
- Die Logger-Komponente ist austauschbar
- Die Logger-Komponente ist plattformunabhängig
- Der Komponentenaustausch ist ausserhalb der IDE und ohne Code-Anpassung (Neukompilation) möglich
- Es können mehrere Instanzen der Logger-Komponente parallel auf den Logger-Server loggen
- Die Logs können dauerhaft auf dem Logger-Server in einem einfachen und lesbaren Textfile gespeichert werden
- Das Textfile enthält mind. die Quelle der Logmeldung, den Zeitstempel der Erstellung, den Zeitstempel beim Erreichen beim Server, den Message-Level und den Message-Text
- Das Schreiben des Textfiles erfolgt serverseitig unter der Verwendung der StringPersistor-Schnittstelle
- Die StringPersistorFile-Komponente persistiert die Logs
- Die Daten werden in strukturierter Form dem Payload-Parameter der StringPersistor-Schnittstelle unter Verwendung des Adapter-Pattern übergeben

4.2.6 Meilensteinziele:

1. Ein Logger-Interface zur Verfügung stellen

2. Die Logger-Komponente implementieren
3. Den Logger-Server implementieren
4. Die Stringpersistor-Komponente implementieren
5. Ereignisse der Applikation (Game of Life) über das Logger-Interface mit der Logger-Komponente (auf dem Logger-Server mittels Stringpersistor) loggen

4.2.7 Wurden die Meilensteinziele erreicht?

1. Ja.
2. Ja.
3. Ja.
4. Ja.
5. Ja.