**Message Logger**

**Projektmanagement-Plan**

[1. Projektorganisation 2](#_Toc498273593)

[1.1. Rollen & Zuständigkeiten 2](#_Toc498273594)

[1.2. Projektstrukturplan 2](#_Toc498273595)

[2. Planung 3](#_Toc498273596)

[2.1. Grobplanung 3](#_Toc498273597)

[2.2. Meilensteine 3](#_Toc498273598)

[2.3. Projektkontrolle 4](#_Toc498273599)

[3. Risikomanagement 4](#_Toc498273600)

[4. Projektunterstützung 5](#_Toc498273601)

[4.1. Konfigurations Items 5](#_Toc498273602)

[4.2. Entwicklungsumgebung 5](#_Toc498273603)

[4.3. Versionskontrolle 6](#_Toc498273604)

[5. Testplan 7](#_Toc498273605)

[5.1. Testdesign & Abläufe 7](#_Toc498273606)

[5.2. Testfälle 7](#_Toc498273607)

[6. Reporting 8](#_Toc498273608)

[6.1. Sprint 1 8](#_Toc498273609)

[6.2. Sprint 2 8](#_Toc498273610)

Versionen:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Rev. | Datum | Autor | Bemerkungen | Status |
| 0.1 | 25.09.2017 | Melvin Werthmüller | 1. Entwurf | Fertig |
| 0.2 | 29.09.2017 | Christopher Christensen | 2. Rollen | Fertig |
| 0.3 | 09.10.2017 | Melvin Werthmüller | Projektunterstützung | Fertig |
| 0.4 | 10.10.2017 | Lukas Arnold, Melvin Werthmüller, Christopher Christensen | Rahmenplan | Fertig |
| 0.5 | 16.10.2017 | Valentin Bürgler | Projektkontrolle | Fertig |
| 0.6 | 17.10.2017 | Christopher Christensen, Melvin Werthmüller, Lukas Arnold | Testplan | Fertig |
| 0.7 | 02.11.2017 | Valentin Bürgler | Ergänzungen zu allen Kapiteln | Fertig |
| 0.8 | 04.11.2017 | Valentin Bürgler | Überarbeitung | Fertig |
| 0.9 | 05.11.2017 | Valentin Bürgler | Bereit für Zwischenabgabe | Fertig |
| 1.0 | 12.11.2017 | Valentin Bürgler | Nachbesserungen Zwischenabgabe | In Bearbeitung |

1. Projektorganisation

## Rollen & Zuständigkeiten

|  |  |
| --- | --- |
| **Name** | **Aufgabe** |
| Christopher Christensen | ProductOwner, Terminplanung |
| Valentin Bürgler | Scrum-Master, Dokumentation |
| Lukas Arnold | Delegierter Interface-Komitee, Code-Master |
| Melvin Werthmüller | Reporting, Projektleiter |

## Projektstrukturplan

Der Projektstrukturplan ist objektorientiert gegliedert in die folgenden Arbeitspakete:

|  |  |
| --- | --- |
| **Arbeitspakete** | **Hauptverantwortlich** |
| LoggerServer | Melvin Werthmüller |
| LoggerComponent | Lukas Arnold & Melvin Werthmüller |
| StringPersistor | Christopher Christensen |
| Logger Interface | Lukas Arnold |
| LoggerSetup Interface | Lukas Arnold |
| StringPersistor Interface | Lukas Arnold |
| GameOfLife | Valentin Bürgler |
| LogLevel | Valentin Bürgler |
| LogMessage | Valentin Bürgler |

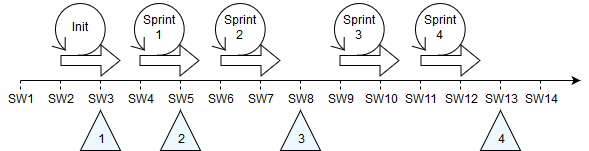
1. Planung

## Grobplanung

|  |  |
| --- | --- |
| Projektstart | SW 2 25.09.2017 15:40 Uhr |
| Projektabschluss | SW 13 11.12.2017 18:00 Uhr |
| Iterationen | 4 Sprints: S1, S2, S3, S4 |
| Projektphasen | Vorprojekt SW 1 18.09.2017 - 24.09.2017  Init SW 2 - SW 3 25.09.2017 - 08.10.2017  S1 SW 4 - SW 5 09.10.2017 - 22.10.2017  S2 SW 6 - SW 7 23.10.2017 - 05.11.2017  S3 SW 9 - SW 10 13.11.2017 - 26.11.2017  S4 SW 11 - SW 12 27.11.2017 - 10.12.2017  Ende SW 13 11.12.2017 |

## Meilensteine

Es wurden vier Meilensteine definiert. Folgende Grafik zeigt, wann diese erreicht werden. Auf dem Zeitstrahl sind die Semesterwochen gekennzeichnet. Darüber sind Initialisierungsphase und die vier Sprintphasen abgebildet. Darunter sind an den entsprechenden Zeitpunkten die Meilensteine gesetzt.



|  |  |  |
| --- | --- | --- |
| Meilenstein | Beschreibung | Zeitpunkt |
| 1 | Organisation der Gruppe ist definiert (SoDa-Rollen); erste Risikoliste, Produktbacklog und Sprintplanung für Sprint 1 liegen vor und sind im PMP dokumentiert. | SW3 |
| 2 | Dokumentationsplan. Liste der Konfigurations-Items. Spezifikation der drei Elemente für das  Systemtesting einschliesslich der Definition des Vorgehens liegt vor.  Entwicklung Sprint 1 abgeschlossen. Code wird in GitLab verwaltet und laufend integriert. Erste (geforderte) Unit-Tests laufen erfolgreich. Sprint 2 geplant (SprintBacklog). | SW5 |
| 3 | Demonstration / Präsentation (Zwischenabgabe Mo. 6./Di. 7. Nov. 2017)  Sprint 2 abgeschlossen. Architektur ist festgelegt und exemplarisch dokumentiert. Release 1 ist lauffähig und kann demonstriert werden. Sprint 3 ist geplant (SprintBacklog).  Vorgängig Abgabe der Dokumentation & Projektcontrolling (elektronisch) => So. 5. Nov. 2017, 18:00 ILIAS Briefkasten,  Peer Review ist organisiert (personell und zeitlich). | SW8 |
| 4 | Sprint 4 abgeschlossen. Nachgeführte Softwarespezifikation liegt vor und ist reviewed. Alle Komponenten sind lauffähig und können demonstriert werden. Die Interoperabilität der Logger-Komponente kann demonstriert werden. Demonstration / Präsentation (Schlussabgabe Di. 12. Dez. 2017)  Vorgängig Abgabe der Dokumentation (elektronisch + Papier).  => Mo. 11. Dez. 2017, 18:00 ILIAS Briefkasten. | SW13 |

## Projektkontrolle

Der Soll/Ist Vergleich wird mittels Zeiterfassung der Work-Items ermittelt. Damit man erkennen kann, ob das Projekt planungsgemäss fortschreitet, verwendet man folgende Tools:

- Sprint Planning

- Daily Scrum

- Sprint Review

- Sprint Retrospective

1. Risikomanagement

Folgende Tabelle listet die für uns relevanten Risiken auf. Die Eintrittswahrscheinlichkeit wird mit der Skala von 1 (unwahrscheinlich) bis 10 (sehr wahrscheinlich) eingestuft. Das Schadensausmass wird von 1 (kein Schaden) bis 10 (erheblicher Schaden) eingeschätzt. Das Produkt aus Eintrittswahrscheinlichkeit und Schadensausmass ergibt einen Wert, der dem Risiko eine Messbarkeit verleiht und einen Vergleich ermöglicht zwischen dem Initial- und dem Restrisiko.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Risikobeschreibung** | **Eintrittswahr-scheinlichkeit** | **Schadens-ausmass** | **Risikowert (E × S)** |
| 1 | Datenverlust | 2 | 8 | 16 |
| 2 | Personenausfall | 5 | 5 | 25 |
| 3 | Änderungen der Requirements | 10 | 5 | 50 |
| 4 | Ausfall des Git-Servers | 2 | 2 | 4 |
| 5 | Ausfall von Ilias | 2 | 1 | 2 |

In der folgenden Tabelle sind die ergriffenen Massnahmen zu den oben beschriebenen Risiken aufgeführt. Weiter werden Eintrittswahrscheinlichkeit und Schadensausmass nach Ergreifen der entsprechenden Massnahme neu geschätzt. Danach werden Inital- und Restrisiko in den letzten beiden Spalten gegenüber gestellt:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Nr.** | **Ergriffene Massnahmen** | **Eintrittswahr-scheinlichkeit** | | **Schadens-ausmass** | | **Risikowert (E × S)** | |
| vorher | nachher | vorher | nachher | vorher | nachher |
| 1 | Änderungen klein halten und regelmässig auf das Git Repository pushen. | 2 | 2 | 8 | 2 | 16 | 4 |
| 2 | Den Ausfall frühzeitig ankündigen und die anfallenden Arbeiten fristgerecht erledigen. | 5 | 5 | 5 | 2 | 25 | 10 |
| 3 | Änderungen der Requirements müssen erwartet werden. Die Code-Basis wird möglichst erweiterbar gehalten. | 10 | 10 | 5 | 1 | 50 | 10 |
| 4 | Im Falle eines Ausfalls von Git wird ausgewichen auf eine alternative Versionsverwaltung wie Mercurial. | 2 | 2 | 2 | 1 | 4 | 2 |
| 5 | Dokumente frühzeitig herunterladen und in eigenes Git Repository einbinden. | 2 | 2 | 1 | 1 | 2 | 2 |

Bemerkungen zu den Risiken mit verbleibendem Restrisikowert von 10:

|  |  |
| --- | --- |
| **Nr.** | **Bemerkung** |
| 2 | Ein Ausfall einer Arbeitskraft bleibt leider ein mögliches Risiko, das sich nicht weiter abfedern lässt. Die anfallende Arbeit muss dann auf die verbleibenden Personen aufgeteilt werden. |
| 3 | Dieses Projekt ist ein Schulprojekt und eine bevorstehende Änderung der Requirments wurde im Vorfeld angedeutet. Dadurch entsteht ein Mehraufwand, welchen wir irgendwie bewältigen müssen. |

1. Projektunterstützung

## Konfigurations Items

* Dokumentation
* Implementation des Loggers, bestehend aus:
  + LoggerComponent
  + LoggerServer
  + LoggerCommon
* Implementation der Stringpersistor-API
* Applikation für Integration des Loggers

## Entwicklungsumgebung

Für die Entwicklung, inkl. dem Testing, wird folgende IntelliJ in folgender Konfiguration verwendet:

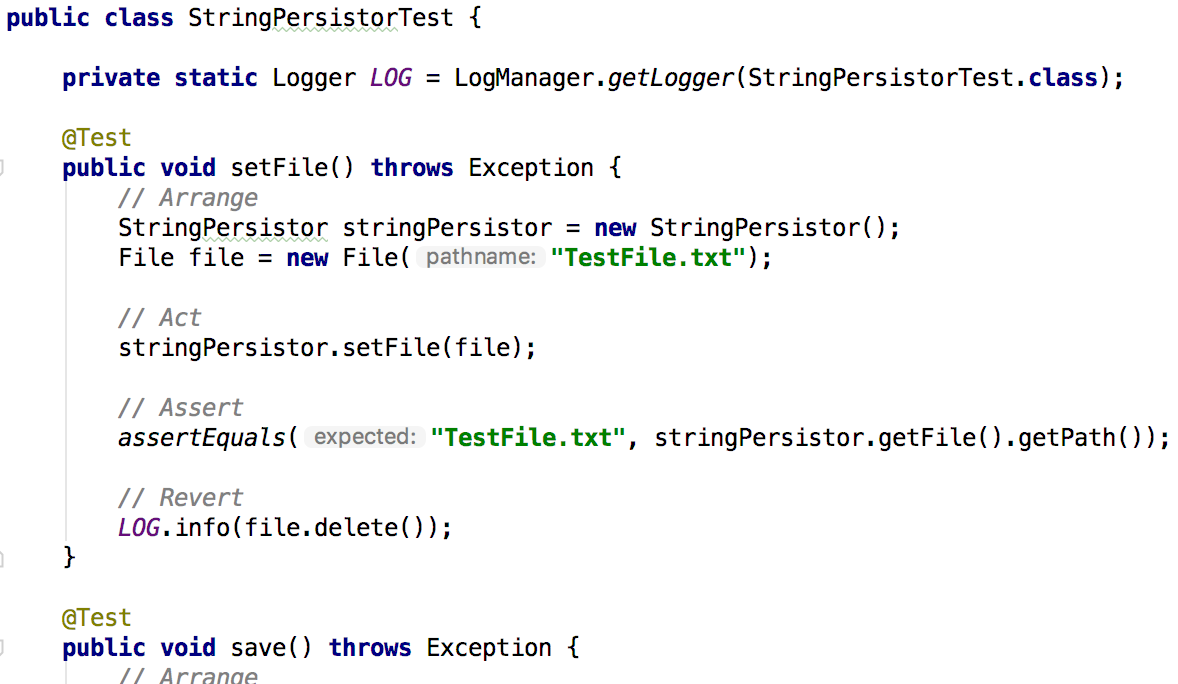
IntelliJ IDEA 2017.2.5

Build #IC-172.4343.14, built on September 26, 2017

JRE: 1.8.0\_152-release-915-b12 amd64

JVM: OpenJDK 64-Bit Server VM by JetBrains s.r.o

Wo es sinnvoll ist Unittests einzusetzen, werden diese nach dem Prinzip „Arrange, Act, Assert (Revert)“ folgendermassen mit JUnit implementiert:



## Versionskontrolle

IntelliJ unterstützt nativ die Verwendung von Versionskontrollwerkzeugen wie Git. Das Projekt wird in den folgenden Git-Repositories verwaltet:

|  |  |
| --- | --- |
| **Verwaltetes Projekt** | **Git-Repository** |
| Logger-Interface für das Logger-Projekt | https://gitlab.enterpriselab.ch/vsk-17hs01/g00-loggerinterface.git |
| Applikation für Integration des Loggers | https://gitlab.enterpriselab.ch/vsk-17hs01/g01-game.git |
| Implementation des Loggers | https://gitlab.enterpriselab.ch/vsk-17hs01/g01-logger.git |
| Implementation der Stringpersistor-API | https://gitlab.enterpriselab.ch/vsk-17hs01/g01-stringpersistor.git |
| Dokumenteverwaltung | https://github.com/christopherchristensen/vsk17 |

1. Testplan

## Testdesign & Abläufe

Um die Funktionalität des Systems zu überprüfen werden die 4 folgenden Tests manuell durchgeführt. Zur Überprüfung der einzelnen Komponenten werden JUnit-Tests verwendet.

## Testfälle

Case 1: Good Case

- Server starten

- Game starten

- Mehrere Aktionen vornehmen

- Log-Einträge auf dem Server überprüfen

Case 2: Internetverbindung zwischendurch unterbrochen

- Server starten

- Game starten

- Mehrere Aktionen vornehmen

- Internetverbindung unterbrechen

- Mehrere Aktionen vornehmen

- Internetverbindung wiederherstellen

- Log-Einträge auf dem Server überprüfen (Aktionen während Verbindungsunterbrechung)

Case 3: Sehr langsame Internetverbindung (Timeouts)

- Server starten

- Game starten

- Mehrere Aktionen vornehmen

- Internetverbindung verlangsamen

- Mehrere Aktionen vornehmen

- Log-Einträge auf dem Server überprüfen

Case 4: Good, Ersetzen der Logger Komponenten, Nochmals Testen

- Server starten

- Game starten

- Mehrere Aktionen vornehmen

- Log-Einträge auf dem Server überprüfen

- Game beenden

- Logger-Komponente austauschen

- Game starten

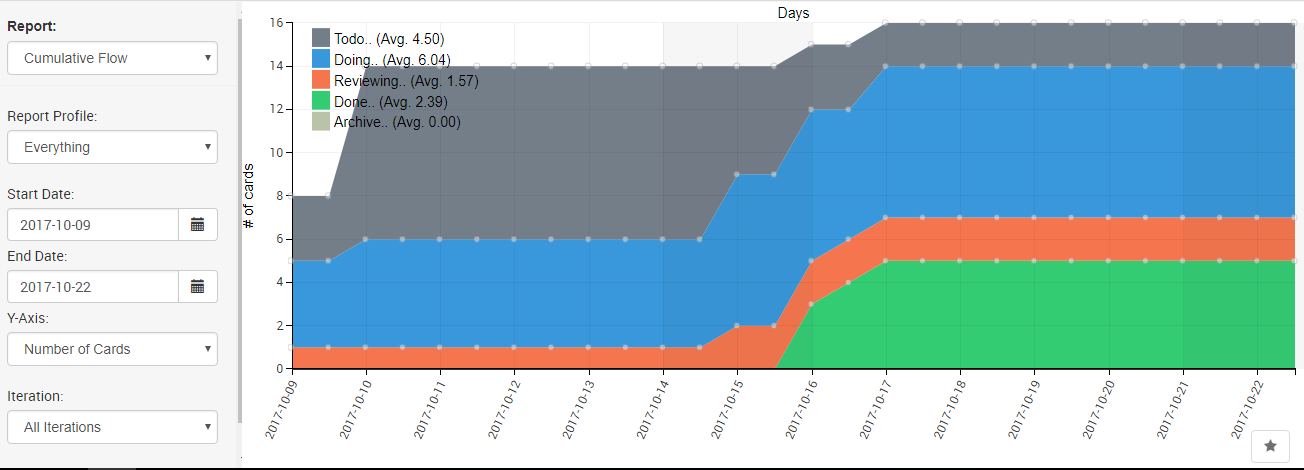
- Mehrere Aktionen vornehmen

- Log-Einträge auf dem Server überprüfen

1. Reporting

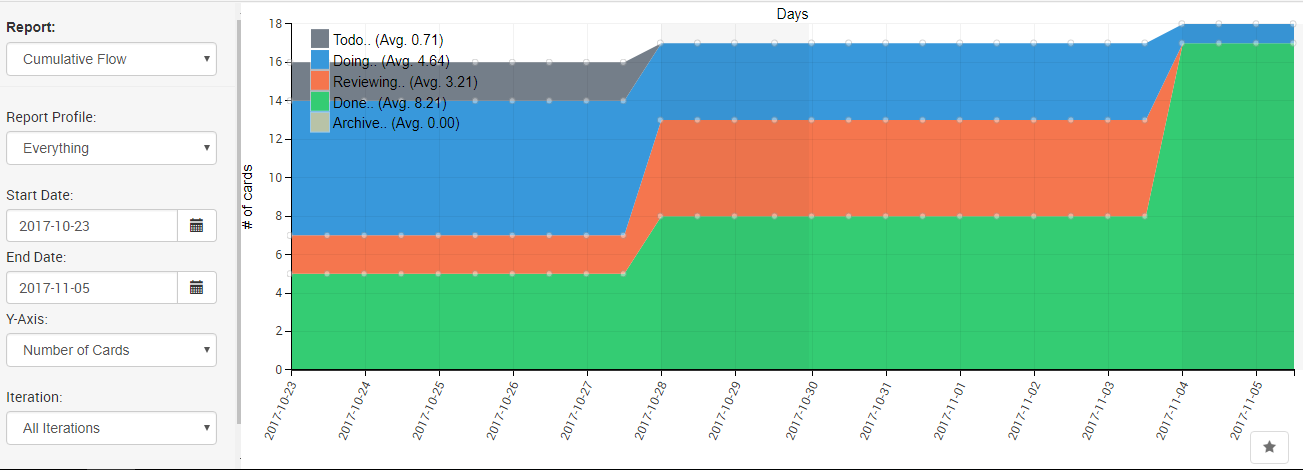
Leider ist das Schätzen der Aufwände und das Führen des Sprintplans in den ersten zwei Sprints nicht reibungslos abgelaufen. Es wurden nicht alle Items geschätzt, die tatsächlichen Aufwände wurden nur selten auf ScrumDo nachgeführt, die Daily Scrum Meetings konnten nicht regelmässig stattfinden und der Product Owner war zeitweilen vakant. Diese Faktoren lassen das Reporting für Sprint 1 und 2 nun etwas mager wirken. Wir haben uns daher im zweiten Sprint Review entschieden, das Cumulative Flow Diagram für diese Sprints zu verwenden, um den Projektfortschritt zu visualisieren. Diese zeigen die Zustände aller Work-Items in einem bestimmbaren Zeitrahmen.

## Sprint 1



Die Abbildung zeigt alle Work-Items über den Zeitraum vom 09.10.2017 bis 22.10.2017. Man kann erkennen, dass die Work-Items nicht alle von Anfang an erfasst worden sind, sondern die Sprintplanung erst nach und nach in die Gänge kam. Auch der erste Sprint Review am 19.10.2017 hat daran noch nicht viel geändert. Ausserdem sieht man, dass bei der Sprintplanung zu viele Items auf den ersten Sprint geplant wurden, welche eigentlich in den Product Backlog gehört hätten.

## Sprint 2



Die Abbildung zeigt alle Work-Items über den Zeitraum vom 23.10.2017 bis 05.11.2017. Es sind auch diejenigen darin enthalten, welche nach dem ersten Sprint Review noch als offen befunden wurden (siehe Review Protokoll Sprint 1.pdf). Man erkennt sehr deutlich, dass der zweite Sprint Review am 04.11.2017 durchgeführt worden ist und es dieses Mal besser funktioniert hat. Gegen Ende des zweiten Sprints wurde uns dann auch klar, dass das bisher versäumte Reporting in Zukunft anders anzugehen ist (siehe Review Protokoll Sprint 2.pdf). An dieser Stelle wurde auch der Entscheid gefasst, dass man für das bisherige Reporting nun mit den Mitteln arbeiten muss, die zu Verfügung stehen.