**Message Logger**

**Projektmanagement-Plan**

[1. Projektorganisation 2](#_Toc500161408)

[1.1. Rollen & Zuständigkeiten 2](#_Toc500161409)

[1.2. Projektstrukturplan 2](#_Toc500161410)

[2. Planung 3](#_Toc500161411)

[2.1. Rahmenplan 3](#_Toc500161412)

[2.2. Meilensteine 3](#_Toc500161413)

[2.3. Projektkontrolle 4](#_Toc500161414)

[3. Risikomanagement 5](#_Toc500161415)

[4. Konfigurationsmanagement 6](#_Toc500161416)

[4.1. Konfigurations Items 6](#_Toc500161417)

[4.2. Releases 6](#_Toc500161418)

[4.3. Entwicklungsumgebung 6](#_Toc500161419)

[4.4. Versionskontrolle 7](#_Toc500161420)

[5. Testing 8](#_Toc500161421)

[5.1. Testplan 8](#_Toc500161422)

[5.2. Testdesign 8](#_Toc500161423)

[5.3. Testfälle 9](#_Toc500161424)

[6. Reporting 11](#_Toc500161425)

[6.1. Sprint 1 11](#_Toc500161426)

[6.2. Sprint 2 11](#_Toc500161427)

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 | Fertig |
| 1.1 | 27.11.2017 | Valentin Bürgler | Anpassungen zu Aufgabenstellung V2 | Fertig |
| 1.2 | 04.12.2017 | Valentin Bürgler | Kapitel 4 eingefügt | Fertig |

1. Projektorganisation

## Rollen & Zuständigkeiten

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

## Projektstrukturplan

Der Projektstrukturplan ist objektorientiert gegliedert in die folgenden Arbeitspakete:

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

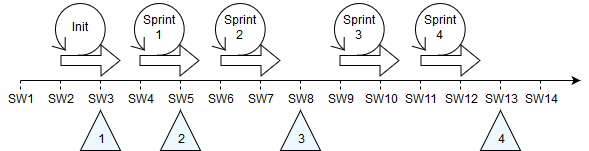
1. Planung

## Rahmenplan

|  |  |
| --- | --- |
| **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 | Auf Änderungen der Requirements wird nicht rechtzeitig reagiert | 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 gegenübergestellt:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **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 | 1 | 8 | 2 | 16 | 2 |
| 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 | 5 | 5 | 1 | 50 | 5 |
| 4 | Im Falle eines Ausfalls von Git wird auf eine alternative Versionsverwaltung wie Mercurial. Ausgewichen. | 2 | 2 | 2 | 1 | 4 | 2 |
| 5 | Dokumente frühzeitig herunterladen und in eigenes Git Repository einbinden. | 2 | 2 | 1 | 1 | 2 | 2 |

1. Konfigurationsmanagement

## Konfigurations Items

* Produktmanagement Plan
* Systemspezifikation
* LoggerInterface, vorgegeben durch das Interface-Komitee
* LoggerComponent
* LoggerServer
* LoggerCommon
* StringPersistorFile
* GameOfLife, die Applikation für die Integration des Loggers
* LoggerViewer

## Releases

|  |  |  |
| --- | --- | --- |
| **Konfigurations Items** | **Release 1, Zwischenabgabe** | **Release 2, Schlussabgabe** |
| Produktmanagement Plan | 0.9 |  |
| Systemspezifikation | 1.6 |  |
| LoggerInterface | 1.2 |  |
| LoggerComponent | 1.0 |  |
| LoggerServer | 1.0 |  |
| LoggerCommon | 1.0 |  |
| StringPersistorFile | 1.0 |  |
| GameOfLife | 1.0 |  |
| LoggerViewer | - |  |

## 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

## 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 (LoggerComponent, LoggerServer und LoggerViewer) | 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 |
| Deploymentverwaltung | https://gitlab.enterpriselab.ch:vsk-17hs01-g01/deployment.git |

1. Testing

## Testplan

Für sämtliche der implementierten Komponenten sind Komponententests zu entwickeln, die automatisiert ausgeführt werden.

Um die Integration der Komponenten zu prüfen werden Integrationstests durchgeführt für folgende Konfigurationsitems:

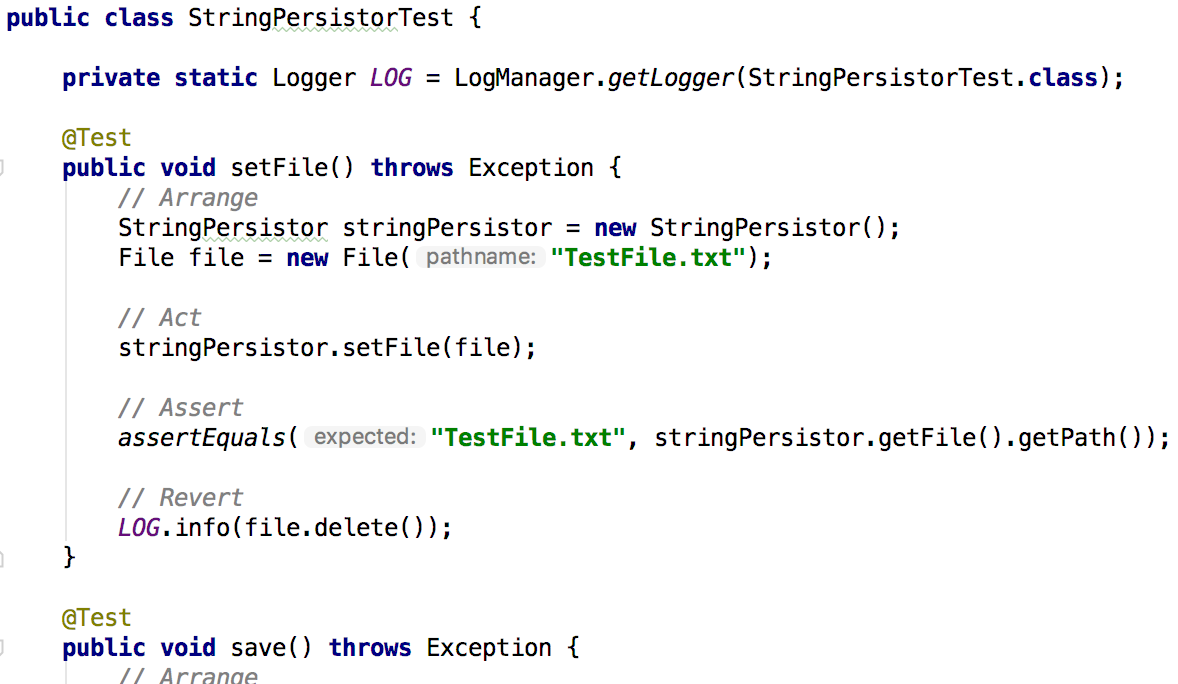
* LoggerComponent
* LoggerServer
* StringPersistorFile
* LoggerViewer

Um die Funktionalität der Systems sicher zu stellen werden Systemtests für folgendes Konfigurationsitem durchgeführt:

* GameOfLife

## Testdesign

Zur automatisierten Überprüfung der einzelnen Komponenten werden JUnit-Tests verwendet. Diese werden nach dem Prinzip „Arrange, Act, Assert (Revert)“ folgendermassen implementiert:



Die Integrations- und Systemtests setzen voraus, dass alle beteiligten Komponenten sämtliche Unittests bestanden haben.

## Testfälle

Um die Integration der Komponenten zu überprüfen werden die folgenden Test-Cases manuell durchgeführt:

|  |  |
| --- | --- |
| Case 1: Logger startet v. 1.0 | |
| Beschreibung | Der Logger kann anhand des Fully Qualified Class Name in der Datei client.properties instantiiert werden |
| Vorbedingung | 1. client.properties liegt vor 2. client.properties enthält den Fully Qualified Class Name einer Klasse, die das Interface LoggerSetup implementiert |
| Erwartetes Ergebnis | Keine Fehlermeldung; der Logger wird instanziiert |
| Testablauf | 1. StandaloneGameOfLive oder GameOfLife starten |

|  |  |
| --- | --- |
| Case 2: Logger loggt lokal v. 1.0 | |
| Beschreibung | Der Logger startet mit dem Spiel und loggt lokal Einträge |
| Vorbedingung | 1. Case 1 2. Server ist nicht gestartet |
| Erwartetes Ergebnis | Es wurden neue Log-Einträge im korrekten Format ins lokal erstellte Logfile.txt geloggt |
| Testablauf | 1. StandaloneGameOfLive oder GameOfLife starten 2. Lokales Logfile.txt öffnen und prüfen, ob es neue Einträge enthält |

|  |  |
| --- | --- |
| Case 3: Logger loggt remote v. 1.0 | |
| Beschreibung | Der Logger startet mit dem Spiel und loggt auf dem Server Einträge |
| Vorbedingung | Case 1 |
| Erwartetes Ergebnis | Der Logger hat neue Log-Einträge im korrekten Format in ein Logfile.txt auf dem Server geloggt |
| Testablauf | 1. Server starten 2. StandaloneGameOfLive oder GameOfLife starten 3. Logfile.txt auf Server öffnen und prüfen, ob es einen neuen Eintrag auf Log-Level «LogLevel.INFO» mit der Nachricht "Initializing UI..." enthält |

|  |  |
| --- | --- |
| Case 4: Server unterstützt mehrere Clients gleichzeitig v. 1.0 | |
| Beschreibung | Der Logger-Server kann mehrere Clients, Messages gleichzeitig verarbeiten |
| Vorbedingung | Case 3 |
| Erwartetes Ergebnis | Es liegen von beiden Clients neue Log-Einträge korrekten Format im Logfile.txt auf dem Server vor |
| Testablauf | 1. Server starten 2. StandaloneGameOfLive oder GameOfLife auf ersten Client starten 3. StandaloneGameOfLive oder GameOfLife auf zweitem Client starten 4. Logfile.txt auf Server öffnen und prüfen, ob es von beiden Clients jeweils einen neuen Eintrag auf Log-Level «LogLevel.INFO» mit der Nachricht "Initializing UI..." enthält |

|  |  |
| --- | --- |
| Case 5: Viewer kann Messages von Server anzeigen v. 1.0 | |
| Beschreibung | Logger Viewer kann Log-Messages des Servers darstellen |
| Vorbedingung | Case 3 |
| Erwartetes Ergebnis | Alle Log-Messages werden im Viewer korrekt dargestellt |
| Testablauf | 1. Server starten 2. Viewer starten 3. StandaloneGameOfLive oder GameOfLife starten 4. Prüfen, ob Viewer Log-Messages anzeigt |

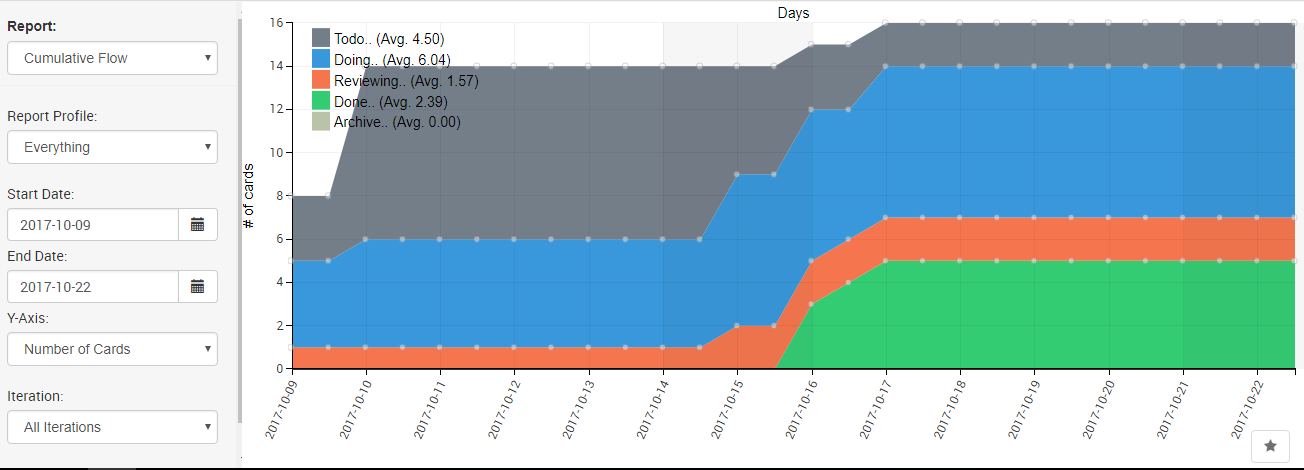
|  |  |
| --- | --- |
| Case 6: Logger kann mit Verbindungsunterbruch umgehen v. 1.0 | |
| Beschreibung | Verbindungsunterbruch nach erfolgtem Verbindungsaufbau wird über lokales Logfile.txt überbrückt. Nach Wiederherstellen der Verbindung werden die lokalen LogMessages an den Server nachgereicht und der Logger loggt wieder auf den Server. |
| Vorbedingung | 1. Case 2 2. Case 3 |
| Erwartetes Ergebnis | Alle Log-Messages werden im Viewer korrekt dargestellt; Es liegt jeweils eine korrekte Log-Message vor für sowohl Verbindungsverlust wie Verbindungswiederaufbau |
| Testablauf | 1. Server starten 2. StandaloneGameOfLive oder GameOfLife starten 3. Internetverbindung trennen 4. Die Shape «Exploder» im Spiel auswählen 5. Internetverbindung wiederherstellen 6. Logfile.txt auf Server öffnen und prüfen, ob der Eintrag auf Log-Level «LogLevel.INFO» mit der Nachricht "Shape Exploder was selected." enthält |

|  |  |
| --- | --- |
| Case 7: Mehrere Viewer können Messages von Server anzeigen v. 1.0 | |
| Beschreibung | Logger Viewer kann Log-Messages des Servers darstellen |
| Vorbedingung | Case 5 |
| Erwartetes Ergebnis | Beide Viewer können sich mit dem Server verbinden; Alle Log-Messages werden in beiden Viewer Instanzen korrekt dargestellt |
| Testablauf | 1. Server starten 2. Ersten Viewer starten 3. Zweiten Viewer starten 4. StandaloneGameOfLive oder GameOfLife starten 5. Prüfen, ob beide Viewer Log-Messages anzeigen |

1. Reporting

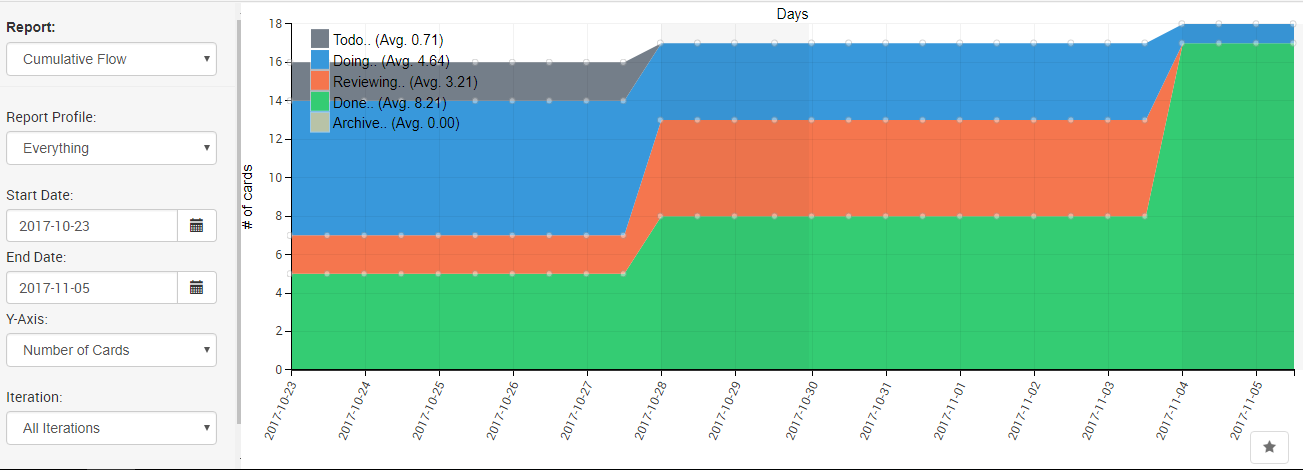
Leider ist das Schätzen der Aufwände und das Führen des Sprintplans in den ersten zwei Sprints nicht reibungslos abgelaufen. Es wurden nicht alle Items geschätzt, die tatsächlichen Aufwände wurden nur selten auf ScrumDo nachgeführt, die Daily Scrum Meetings konnten nicht regelmässig stattfinden und der 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.