[*Team 1*]

[*Christopher Christensen*] [*Valentin Bürgler*] [*Lukas Arnold*] [*Melvin Werthmüller*]

Message Logger

Projektmanagement-Plan

Versionen:

Rev.	Datum	Autor	Bemerkungen	Status
0.1	25.09.2017	MW	1. Entwurf	Fertig
0.2	29.09.2017	CC	2. Rollen	Fertig
0.3	09.10.2017	MW	Projektunterstützung	Fertig
0.4	10.10.2017	LA, MW, CC	Rahmenplan	Fertig
0.5	16.10.2017	VB	Projektkontrolle	Fertig
0.6	17.10.2017	LA, MW, CC	Testplan	Fertig
0.7	02.11.2017	VB	Ergänzungen zu allen Kapiteln	Fertig
0.8	04.11.2017	VB	Überarbeitung	Fertig
0.9	05.11.2017	VB	Bereit für Zwischenabgabe	Fertig
1.0	12.11.2017	VB	Nachbesserungen Zwischenabgabe	Fertig
1.1	27.11.2017	VB	Anpassungen zu Aufgabenstellung V2	Fertig
1.2	04.12.2017	VB	Kapitel 4 eingefügt	Fertig
1.3	05.12.2017	MW, VB	Testfall 2 obsolet, Streamlining Namensgebung	Fertig
2.0	10.12.2017	VB	Überarbeitung und Abschluss	Fertig

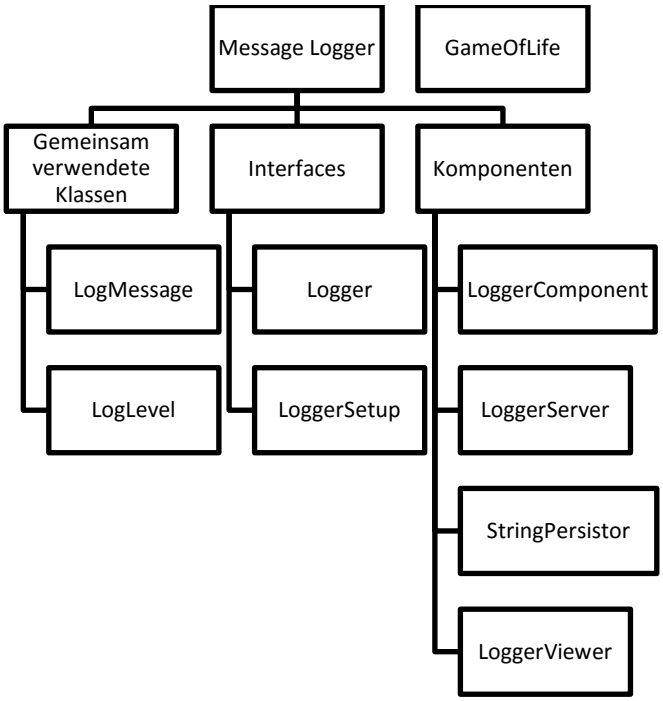
1. Projektorganisation

1.1. Rollen & Zuständigkeiten

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

1.2. 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
GameOfLife	Valentin Bürgler
LogLevel	Valentin Bürgler
LogMessage	Valentin Bürgler
LoggerViewer	Lukas Arnold

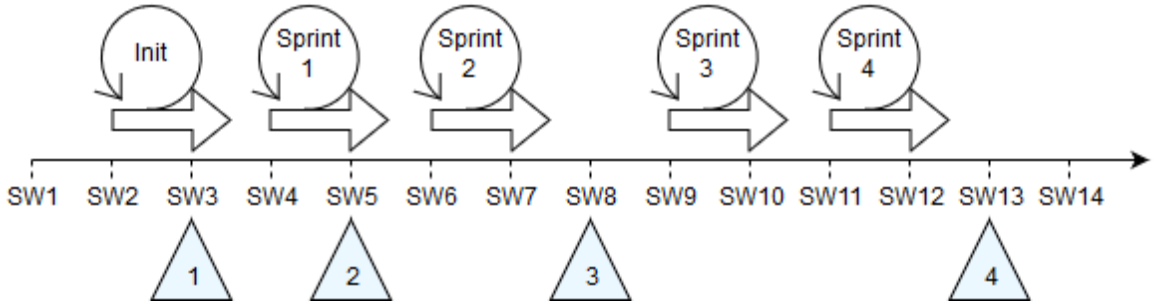
2. Planung

2.1. 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

2.2. 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

2.3. Projektkontrolle

Der Soll/Ist Vergleich wird mittels Zeitschätzung und -erfassung der Work-Items ermittelt. Damit man erkennen kann, ob das Projekt planungsgemäss fortschreitet, werden folgende Methoden angewandt:

- Sprint Planning
- Daily Scrum
- Sprint Review
- Sprint Retrospective

3. 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

4. Konfigurationsmanagement

4.1. Konfigurations-Items

- Produktmanagementplan
- Systemspezifikation
- LoggerInterface, vorgegeben durch das Interface-Komitee
- LoggerComponent
- LoggerServer
- LoggerCommon
- StringPersistor
- StringPersistorAPI, vorgegeben durch Aufgabenstellung
- GameOfLife, vorgegeben durch Aufgabenstellung
- LoggerViewer

4.2. Releases

Konfigurations Items	**Release 1, Zwischenabgabe**	**Release 2, Schlussabgabe**
Produktmanagementplan	0.9	2.0
Systemspezifikation	1.6	4.1
LoggerInterface	1.0	1.0
LoggerComponent	1.0	2.0
LoggerServer	1.0	2.0
LoggerCommon	1.0	2.0
StringPersistor	1.0	1.0
StringPersistorAPI	4.0	4.0
GameOfLife	1.0	1.0
LoggerViewer	(noch nicht vorliegend)	2.0

4.3. 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

4.4. Versionskontrolle

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

Verwaltetes Projekt	**Git-Repository**
LoggerInterface für das Logger-Projekt	https://gitlab.enterpriselab.ch/vsk-17hs01/g00-loggerinterface.git
GameOfLife, die 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 StringPersistorAPI	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

5. Testing

5.1. Testplan

Für die eigens implementierten Komponenten liegen Komponententests vor, die automatisiert ausgeführt werden.
Um die Integration der Komponenten zu prüfen werden Integrationstests durchgeführt für folgende Konfigurationsitems:

- LoggerComponent
- LoggerServer
- StringPersistor
- LoggerViewer
- GameOfLife

5.2. Testdesign

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

```
public class StringPersistorTest {

    private static Logger LOG = LogManager.getLogger(StringPersistorTest.class);

    @Test
    public void setFile() throws Exception {
        // Arrange
        StringPersistor stringPersistor = new StringPersistor();
        File file = new File( pathname: "TestFile.txt");

        // Act
        stringPersistor.setFile(file);

        // Assert
        assertEquals( expected: "TestFile.txt", stringPersistor.getFile().getPath());

        // Revert
        LOG.info(file.delete());
    }

    @Test
    public void save() throws Exception {
        // Arrange
```

Die Integrationstests setzen voraus, dass alle beteiligten Komponenten sämtliche Unittests bestanden haben. Das Bestehen der jeweiligen Integrationstests dient während der Entwicklung auch als Abnahmekriterium.

5.3. Testfälle

Um die Integration der Komponenten zu überprüfen werden die folgenden Testfälle manuell durchgeführt:

Case 1: Logger startet	v. 1.0
Beschreibung	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 obsolet.

Case 3: Logger loggt remote	v. 1.0
Beschreibung	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	LoggerServer 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	LoggerViewer 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. 1.1
Beschreibung	Verbindungsunterbruch nach erfolgtem Verbindungsaufbau wird über lokales LogfileTMP.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 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

5.4. Meilenstein 3 - Zwischenabgabe

Die Artefakte (Dokumentation, Code, Tests, etc.) werden bei der Zwischenabgabe durch die Dozenten Roger Diehl, Roland Gisler und Martin Jud auf Vollständigkeit und fachliche Richtigkeit geprüft. Ausserdem sind die Muss-Features Nr. 1 – 9 realisiert [1].

5.5. Peer Review

Die Artefakte (Dokumentation, Code, Tests, etc.) werden von Team 2, bestehend aus Dao Trong Viet, Kressibucher Michael, Summermatter Oliver und Wicki Sandro überprüft darauf, dass sie den Anforderungen für die Schlussabgabe fachlich genügen und vollständig sind[2]. Die Muss-Features Nr. 1 – 14 sind realisiert[3].

[1] Siehe „2.3 Muss-Features bei Zwischenabgabe“ in „VSK_LoggerProjektauftrag_V2.pdf“ auf ILIAS
[2] Siehe „Review-Bericht.pdf“ im Anhang
[3] Siehe „2.4 Muss-Features bei Schlussabgabe“ in „VSK_LoggerProjektauftrag_V2.pdf“ auf ILIAS

6. Anhänge

- Protokolle der Sprintreviews 1-4
- Review-Protokoll aus dem Peer Review
- Sprint Retrospektive