sniffdatel

SW Engineering Projekt FS 2016

Projektplan

David Meister, Giorgio Vincenti, Samuel Krieg, Andreas Stalder

15. April 2016

1 Änderungsgeschichte

Datum	Version	Änderung	Autor
04.03.16	1.0	Erstellung erster Version	Alle
07.03.16	1.1	Korrektur nach erstem Review	sk/dm
10.03.16	1.1	Arbeitspakete und Projektorganisation	gv
15.04.16	1.2	Aktualisierung Mgmt Abläufe	gv

Tabelle 1: **Änderungsgeschichte**

Inhaltsverzeichnis

2 Einführung

2.1 Zweck

Dieses Dokument stellt den Projektplan für unser Engineering-Projekt dar, es dient zur Planung, Steuerung und Kontrolle.

2.2 Gültigkeitsbereich

Dieses Dokument ist über die gesamte Projektdauer gültig. Es wird in späteren Iterationen angepasst. Somit ist jeweils die neuste Version des Dokuments gültig und alte Versionen sind obsolet.

2.3 Referenzen

jNetPcap
http://jnetpcap.com/

Java Media Framework 2.0
http://download.oracle.com/otndocs/jcp/7273-jmf-2.0-fr-spec-oth-JSpec/

RTP A Transport Protocol for Real-Time Applications (RFC)
https://tools.ietf.org/html/rfc1889

SIP Session Initiation Protocol (RFC)
https://tools.ietf.org/html/rfc3261

SDP Session Description Protocol (RFC)
https://tools.ietf.org/html/rfc4566

3 Projekt Übersicht

sniffdatel ermöglicht das Aufzeichnen und Abspielen von Voice over IP Paketen in Echtzeit. Wireshark bietet seit langem die Möglichkeit, den Netzwerkverkehr auf der Karte aufzuzeichnen, RTP Streams zu filtern und diese abzuspielen. Wireshark ist jedoch nicht in der Lage, RTP Datenpakete in Echtzeit wiederzugeben. sniffdatel filtert RTP Streams aus dem mitgeschnittenen Netzwerkverkehr, stellt die Streams auf einem GUI dar und spielt sie nach Wunsch ab.

3.1 Zweck und Ziel

Im Engineering-Projekt sollen die Teammitglieder das im Software-Engineering 1 Modul erworbene Wissen praktisch anwenden. Es soll ein vollständiges Softwareprodukt von der Anforderungsspezifikation bis zum getesteten Code entwickelt und dokumentiert werden. Mittels unterstützenden Werkzeugen wie z.B. Redmine und git soll das Teamverhalten erlernt und gefördert werden.

Wir Teammitglieder bekunden grosses Interesse in den Informatikbereichen Computernetze und Informationssicherheit, deshalb war für uns klar, ein Softwareprodukt in diesem Bereich zu entwickeln.

Wir konnten uns an Übungslektionen erinnern, in denen wir RTP Datenpakete mit Wireshark aufgezeichnet und abgespielt haben. Dies war einerseits faszinierend, andererseits mühsam und wenig intuitiv. Uns ist die Idee gekommen, ein Netzwerksniffer nur für den Zweck, VoIP Pakete aufzuzeichnen und abzuspielen, zu entwickeln.

3.2 Lieferumfang

Dieses Projekt umfasst die fertige Software, allfällige Handbücher, Prototypen und Präsentationen.

3.3 Annahmen und Einschränkungen

Es wird von einem Umfang von geschätzten 120 Stunden pro Teammitglied ausgegangen. Erweist sich die geplante Zeit als zu knapp, oder ein Feature als nicht realisierbar, so wird dies in Absprache mit dem Betreuer gegebenenfalls weggelassen.

4 Projektorganisation

Das Projektteam besteht aus vier gleichgestellten Mitgliedern, es wird bewusst auf einen Projektleiter verzichtet. Sämtliche Entscheidungen werden als Team gefällt. Das Projektteam wird von Andreas Steffen betreut.

4.1 Organisationsstruktur

Vorname	Name	E-Mail	Veratwortlich für
Andreas	Stalder	astalder@hsr.ch	Usability- und Abnahmetest Verantwortlicher
David	Meister	dmeister@hsr.ch	Systemtest Verantwortlicher
Giorgio	Vincenti	gvincent@hsr.ch	Redmine Verantwortlicher
Samuel	Krieg	skrieg@hsr.ch	Git und Dokumentvorlagen Verantwortlicher

Tabelle 2: **Teammitglieder**

Primäre Ansprechsperson für organisatorische Belange ist David Meister.

4.2 Externe Schnittstellen

Das Projekt wird von Andreas Steffen betreut und benotet. Für Usability Tests werden weitere externe Personen involviert.

5 Management Abläufe

5.1 Kostenvoranschlag

Der Projektstart ist am Montag den 22. Februar 2016.
Die Projektdauer beträgt 15 Wochen, und das Projektende ist am Freitag den 3. Juni 2016.

Während diesen 15 Wochen sind 120 Arbeitsstuden pro Projektmitglied eingeplant. Das entspricht pro Mitglied eine Arbeitszeit von acht Stunden pro Woche. Dies ergibt einen totalen Aufwand von 480 Stunden.

Die wöchentliche Arbeitszeit von acht Stunden kann bei Verzug oder bei unerwarteten Problemen auf maximal 12 Stunden erhöht werden.

Es sind gegenwärtig keine Absenzen während dieser Zeit geplant.

5.2 Zeitliche Planung

Die Zeitplanung und die Verwaltung der Arbeitspakete erfolgt in Redmine. Diese wird während dem Projekt laufend aktualisiert. Die im Redmine erzeugten Tickets dienen als Arbeitspakete. Diese werden einer, ebenfalls im Redmine hinterlegten, Iteration zugewiesen. Anhand von diesen Daten ist ein übersichtlicher Zeitplan ersichtlich. Um einen Überblick über den aktuellen Zeitplan zu erhalten, erfolgt der Zugriff auf das Gantt-Diagram via URL: http://152.96.56.43/redmine/projects/ep2016_realtimeplayer/issues/gantt Die Projektmitglieder tragen jeweils die investierte Zeit am Abend, in das zugewiesene Ticket ein.

5.2.1 Phasen

Das Projekt wird in vier Phasen unterteilt: Inception, Elaboration, Construction und Transition.

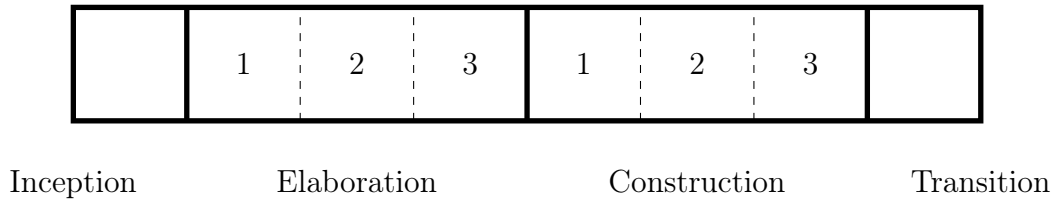

Abbildung 1: **Phasenplan**

5.2.2 Meilensteine

Das Projekt beinhaltet insgesamt acht Meilensteine.
**Update: MS3 kann zeitlich nicht eingehalten werden, und wurde nach Absprache mit dem Betreuer auf den 18.4.16 verschoben.*

Meilenstein	Beschreibung	Datum
MS1	Review Projektplan	07.03.16
MS2	Review Anforderungen und Analyse	22.03.16
MS3*	Zwischenpräsentation mit Demo eines Architekturprotypen	11.04.16
MS4	Review Architektur und Design	18.04.16
MS5	Präsentation Alpha Version	09.05.16
MS6	Präsentation Beta Version	16.05.16
MS7	Software Version 1.0	23.05.16
MS8	Präsentation und Abgabe	03.06.16

Tabelle 3: **Projekt Meilensteine**

5.2.3 Iterationen

Die Dauer eines Iterationszyklus beträgt jeweils zwei Wochen.

Iteration	Inhalt	Start	Ende
Inception	SW1: Themenwahl, Projektantrag	22.02.2016	29.02.2016
Elaboration1	SW2/3: Projektplan, Diagramme erstellen	01.03.2016	13.03.2016
Elaboration2	SW4/5: Entwurf GUI, Architektur und Design	14.03.2016	27.03.2016
Elaboration3	SW6/7: GUI Abschluss, Prototyp	28.03.2016	11.04.2016
Construction1	SW8/9: Prototypen programmieren	12.04.2016	24.04.2016
Construction2	SW10/11: Prototypen programmieren	25.04.2016	09.05.2016
Construction3	SW12/13 Ausbau Prototypen, Release	10.05.2016	23.05.2016
Transition	SW14/15: Präsentation, Projektabschluss	24.05.2016	03.06.2016

Tabelle 4: **Projekt Iterationen**

5.2.4 Arbeitspakete (Tickets)

Name	Inhalt	Iteration	Wer	Soll	Ist
Projektstart					
Themenwahl	Thema wählen und mit Betreuer besprechen	Inception	Alle	4	6.5
Projektantrag	Projektantrag erstellen	Inception	Alle	2	2.5
Projektplan					
Einführung	Einführungs- und Übersichtkapitel in Projektplan erstellen	Elaboration1	dm	2	2
Organistation	Organisationskapitel in Projektplan erstellen	Elaboration1	dm	1	2
Management Abläufe	Phasen, Iterationen, Meilensteine und Arbeitspakete definieren und in Redmine eintragen	Elaboration1	gv	16	26
Risikomanagement	Risikomanagement Kapitel erstellen und vorhandene Risiken abschätzen	Elaboration1	dm/as	2	6
Infrastruktur	Infrastruktur Kapitel erstellen	Elaboration1	as	2	1.5
Qualitätsmanagement	Qualitätsmanagement Kapitel erstellen	Elaboration1	sk/as	5	12
Review Projektplan	Projektplan gemäss Review anpassen	Elaboration1	alle	6	6

Name	Inhalt	Iteration	Wer	Soll	Ist
Anforderungen + Analyse					
Use Case Diagramm + fully dressed	Use Case Diagramm erstellen	Elaboration1	sk/gv	5	19.5
Supplementary Spec.	Nicht-funktionale Anforderungen definieren	Elaboration1	dm	3	4.5
Domain Modell	Domain Modell für Software erstellen	Elaboration1	dm/as	5	15.7
SDD	System Sequenz Diagramm für Projekt erstellen	Elaboration1	as	4	3.5
Operation Contracts	Verfassen der Contracts	Elaboration1	gv	2	8.75
Activity Diagramm	Darstellen der Abläufe	Elaboration1	as	4	2.5
Zustandsdiagramm	Darstellen der möglichen Zustände	Elaboration1	sk	4	9.5
Entwurf GUI	GUI Entwurf ausarbeiten	Elaboration2	alle	8	2
Review + Korrektur Anforderungen+Analyse	Ausarbeiten und Verbesserungen gemäss Review Feedback	Elaboration2	alle	8	
Architektur	Definieren der Software-Architektur	Elaboration2	dm/gv	10	
Design					
Design Model	Klassendiagramme erstellen	Elaboration2	as/sk	6	

Name	Inhalt	Iteration	Wer	Soll	Ist
Logische Architektur	Definieren der Architektur im Detail	Elaboration2	as/sk	15	
GUI Design fertigstellen	GUI Entwurf ausarbeiten und fertigstellen	Elaboration3	alle	16	0
Implementation					
Prototyp GUI	Erster Prototyp für GUI programmieren	Elaboration3	sk	26	
Prototyp Session finder	Erster Prototyp für Session Zuordnung programmieren	Construction1	as/sk	18	17.5
Prototyp Packet aufzeichnung	Erster Prototyp für Paket aufzeichnung mittels pcab library	Construction1	as/sk	18	29
Prototyp Audio wiedergabe	Erster Prototp für Audio Verarbeitung und Abspielung programmieren	Construction2	gv/dm	24	63.5
Prototypen zusammenführen	Alle erstellten Prototypen zusammenführen	Construction2	as/dm	10	6
Ausbau GUI	Prototypen GUI ausbauen	Construction3	xx	24	
Ausbau Session finder	Prototypen Session finder ausbauen	Construction3	xx	16	
Ausbau Packet aufzeichnen	Prototypen Aufzeichnung der Pakete ausbauen	Construction3	xx	16	
Ausbau Audio wiedergabe	Prototypen Audio Wiedergabe ausbauen	Construction3	xx	16	
Test + Bugfixing					

Name	Inhalt	Iteration	Wer	Soll	Ist
Usability Tests	Usability Tests erstellen und dokumentieren	Construction3	xx	12	
Systemtest mit Telefon	Software Tests mit kompletter Infrastruktur (Softphones, Netzwerk..)	Construction3	alle	12	
Abnahmetest	Software Abnahmetest-Dokument	Construction 3	xx	12	
Dokumentation					
Benutzeranleitung	Benutzeranleitung für Software erstellen	Transition	xx	8	
Übersicht Q-Massnahmen	Übersicht Dokument für Q-Massnahmen erstellen	Transition	xx	16	
Schlusspräsentation	Schlusspräsentation erstellen + vorbereiten	Transition	alle	40	
Abgabe vorbereiten	Dokumente und Software soweit fertig für Abgabe vorbereiten	Transition	alle	16	
Sitzungen+Dokumente					
Meeting	Wöchentliche Team Meetings	laufend	alle	60	
Meeting mit Betreuer + Reviews	Team Meetings mit Betreuer, Reviews inklusive	laufend	alle	20	
Dokumentvorlagen	Dokumentvorlagen erstellen auf Git	laufend	alle	8	

Name	Inhalt	Iteration	Wer	Soll	Ist

Tabelle 5: **Arbeitspakete**

5.3 Besprechungen

Besprechungen finden wöchentlich jeweils am Montag statt. Eine Besprechung dauert in der Regel 30min und findet in der HSR (meistens Gebäude 1) statt. Bei einer Besprechung wird das weitere Vorgehen, sowie durchgeführte Arbeiten, fällige Arbeiten und auftretende Probleme besprochen. Weiter werden Arbeitspakete verteilt, damit alle Projektmitglieder wissen was zu tun ist.

Als Kommunikationsmittel wird eine Whatsapp Gruppe verwendet.

5.3.1 Reviews

Die Reviews zur Arbeit mit dem Betreuer finden Montags ab 15:00 Uhr statt. Die Reviews werden mit dem Betreuer Andreas Steffen in seinem Büro durchgeführt. Die Dauer eines Reviews ist unterschiedlich und kann start variieren.

6 Risikomanagement

6.1 Risiken

Technische Risiken in der Entwicklung sind im Dokument TechnischeRisiken.xlsx aufgeführt.

6.2 Umgang mit Risiken

Die im Dokument TechnischeRisiken.xlsx aufgeführten Risiken sind in der Zeitplanung nicht speziell vorgesehen. Falls beim Eintreten eines geplanten Risikos ein erhöhter Zeitbedarf entsteht, so muss dies mit hoher Wahrscheinlichkeit mit Mehrarbeit der Teammitglieder kompensiert werden. Falls die nötige Mehrarbeit ausserhalb der Möglichkeiten liegt, so muss in Absprache aller Teammitglieder mit dem Betreuer nach einer anderer Lösung (z.B. Einschränkung von Programmfeatures, etc.) gesucht werden.

7 Infrastruktur

Software	Version	Beschreibung
Eclipse IDE	Mars.2 (4.5.2)	IDE zur Entwicklung von Software. Wird für alle Entwicklungsaufgaben verwendet
JUnit	4.12	Testframework für das Testen von Java-Programmen
EclEmma	2.3.3	Werkzeug, welches die Testabdeckung in Java-Programmen misst
Redmine	3.2.0	Projektmanagementtool
Git	2.7.2	Verteiltes Versionsverwaltungsystem
jitsi	2.8	VoIP Softphone
LaTeX	2	Textsatzsystem
WhatsApp	2.12.14	Teamkommunikation
OneNote	2016	Notizen im Team
Dropbox	3.14.7	Teilen von Dokumenten ausserhalb von Git

Tabelle 6: **Infrastruktur**

8 Qualitätsmassnahmen

Massnamen	Zeitraum	Ziel der Massnahme
Git verwenden	immer	Versionierung und Verhinderung von Datenchaos
Redmine verwenden	immer	Einhaltung von Vorgehen und Zeitplan
Teamsitzung	1h pro Woche	Sicherstellung der erfolgreichen Kommunikation.
Codereviews	nach Abschluss von Ticket	Garantierung guter Codequalität
Styleguide für Code	immer	Code lesbarkeit und Wartungsfreundlichkeit
Tests	in und nach der Programmierphase	Sicherstellung der Funktionalität

Tabelle 7: **Qualitätsmassnahmen**

8.1 Dokumentation

Alle Datein, welche Teil der Dokumentation sind, werden mit Git versioniert. Das Git Repository befindet sich auf GitHub.

8.2 Projektmanagement

Als Projektmanagementsoftware wird Redmine eingesetzt. Es wird nach jeder Arbeitssession oder beim Wechsel einer Arbeit der Aufwand auf das entsprechende Ticket verbucht. Zugriff auf Redmine erfolg über die Url: http://152.96.56.43/redmine/ Um den Zugriff für Betreuungspersonen zu ermöglichen wurde ein Gastbenutzer eingerichtet.

Logindaten Redmine Gastbenutzer:

Login: guest

Password: guest2016

8.3 Entwicklung

Wie die Dokumentation wird auch der Code mit Git versioniert und auf GitHub abgelegt.

8.3.1 Vorgehen

Als Erstes erfolgt die Einarbeitung in das entsprechende Thema. Nach Erstellung eines Konzeptes werden die Features separiert entwickelt. Wurden Reviews und Tests erfolgreich durchgeführt, kann die Zusammenführung erfolgen.

8.3.2 Code Reviews

Damit wir eine Kontrolle über den Code haben, wird jedes Feature von mindestens einer anderen Person betrachtet. Dazu wird wie folgt vorgegangen:
Die zuständige Person entwickelt das vorgesehene Feature und schreibt Tests dazu. Wenn man mit seiner Arbeit zufrieden ist, bekommt das Feature den Status Feedback. All diese Feedback-Tickets werden einmal pro Woche von mindestens einem anderen Teammitglied überprüft. Wenn alles in Ordnung ist, wird das Ticket auf Erledigt gesetzt. Falls ein Fehler gefunden wurde, wird ein Kommentar hinzugefügt und das Ticket bekommt den Status In Bearbeitung.

8.3.3 Code Style Guidelines

In Anlehnung an die Java Code Conventions wurde eine Code Style Guideline definiert, und in der Datei CodeStyleProfile.xml beschrieben. Es ist möglich dieses Definition ins Eclipse einzubinden. Nachfolgend die Quelltextformatierung welche für das Projekt verwendet wird.

```
package mypackage;

import java.util.LinkedList;

public class MyIntStack {

  private final LinkedList fStack;
```

```
  public MyIntStack() {
    fStack = new LinkedList();
  }

  public int pop() {
    return ((Integer) fStack.removeFirst()).intValue();
  }

  public void push(int elem) {
    fStack.addFirst(new Integer(elem));
  }

  public boolean isEmpty() {
    return fStack.isEmpty();
  }
}
```

8.4 Testen

8.4.1 Unit Tests

Um eine hohe Qualität für das gesamte Projekt zu erhalten, werden für alle wichtigen Komponenten/Klassen Unit Tests geschrieben. Dazu verwenden wir JUnit4. So können wir nach jedem Entwicklungsschritt überprüfen, ob die Tests noch funktionieren. Die Tests zu den Klassen werden zum Teil vor dem Programmieren und zum Teil nach dem Programmieren der Klasse gemacht. Damit die Komponente abgenommen wird, muss jeder Test erfolgreich durchlaufen.

8.4.2 Systemtest

Nachdem das Programm die Alpha- und Betaphase erreicht hat, wird jeweils ein Systemtest gemacht. Dafür wird vorher eine Testspezifikation geschrieben. Die Ergebnisse werden in einem Testprotokoll erfasst und durch dieses Protokoll werden Bugreports zu den Tickets hinzugefügt.

8.4.3 Abnahmetest

Sobald das Produkt fertig entwickelt wurde, wird ein Abnahmetest durchgeführt. Dafür wird vorgängig eine Testspezifikation geschrieben. Die Testspezifikation be-

inhaltet die am Anfang besprochenen Anforderungen, sowie auch anderen relevanten Tests.

8.4.4 Usability Test

Nachdem das User Interface funktioniert, werden erste Usability Tests durchgeführt. Es werden mehrere externe Personen einbezogen, welche das Programm auf Usability testen und bewerten. Dazu wird vorgängig ein Testspezifikation mit verschiedenen Bewertungspunkten erstellt. Nachdem der Test abgeschlossen ist, besprechen wir die verschiedenen Punkte im Team und planen, was verändert wird. Die besprochenen Punkte werden danach in das Projekt eingepflegt.

8.4.5 Testabdeckung

Für die Testabdeckung werden wir EclEmma einsetzen. Durch die Tests wollen wir eine möglichst hohe Abdeckung(90%) des nicht trivialen Codes.