

software name

HSR Studienarbeit Network Unit Testing

Projektplan

David Meister, Andreas Stalder

28. September 2016

1 Änderungsgeschichte

Datum	Version	Änderung	Autor
27.09.16	1.0	Erstellung erster Version	dm/as

Tabelle 1: **Änderungsgeschichte**

Inhaltsverzeichnis

1	Änderungsgeschichte	2
2	Einführung	5
2.1	Zweck	5
2.2	Gültigkeitsbereich	5
2.3	Referenzen	5
3	Projektübersicht	5
3.1	Zweck und Ziel	6
3.2	Lieferumfang	6
3.3	Annahmen und Einschränkungen	6
3.4	Organisationsstruktur	6
3.5	Externe Schnittstellen	6
4	Management Abläufe	7
4.1	Zeitbudget	7
4.2	Zeitliche Planung	7
4.2.1	Phasen	7
4.2.2	Meilensteine	8
4.2.3	Iterationen	8
4.2.4	Arbeitspakete (Tickets)	10
4.3	Teammeeting	11
4.3.1	Meeting mit Betreuern	11
5	Risikomanagement	11
5.1	Risiken	11
5.2	Umgang mit Risiken	11
6	Qualitätsmassnahmen	12
6.1	Dokumentation	12
6.2	Projektmanagement	12
6.3	Entwicklung	13
6.3.1	Versionierung	13
6.3.2	Test Driven Development	13

6.3.3	Code Metriken	13
6.3.4	Code Reviews	13
6.3.5	Continuous Integration	13

2 Einführung

2.1 Zweck

Dieses Dokument stellt den Projektplan für unser Studienarbeit 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

Noch keine.

3 Projektübersicht

Nach durchgeführten Changes im Netzwerkbereich wird die gewünschte Funktionalität meist von Hand getestet. In den meisten Fällen geschieht dies durch einfache Tools wie ping oder traceroute. Es wird vom durchführenden Netzwerk Engineer erwartet, dass er selbständig weiss, welche Funktionalität er nach durchgeführten Changes testen muss. Je nach Komplexität der Infrastruktur ist der Scope kaum in den Griff zu kriegen.

In der Software Entwicklung nutzt man schon lange Unit Tests. Optimalerweise werden für Softwareklassen zuerst Unit Tests geschrieben und anschliessend wird der Code implementiert. Somit existiert für jede Klasse und deren Methoden eine Testsammlung, welche bei späteren Changes immer ausgeführt werden kann.

Mit dieser Arbeit wird die Möglichkeit der Nutzung für Netzwerk Unit Tests angestrebt.

3.1 Zweck und Ziel

Die Studienarbeit soll den Nachweis der Problemlösungsfähigkeit unter Anwendung ingenieurmässiger Methoden nachweisen. Entsprechend verfügt die Arbeit über einen konzeptionellen, theoretischen und einen praktischen Anteil.

Mit Unit Tests für Netzwerkinfrastrukturen möchte man ein technisches Hilfsmittel für die Qualitätssicherung in der IT bereitstellen. Ähnlich wie in der Software Entwicklung soll eine Systematik entwickelt werden, um Komponenten im Netzwerk auf ihre Konfiguration und Funktionalität zu testen.

3.2 Lieferumfang

noch nicht definiert

3.3 Annahmen und Einschränkungen

bisher noch keine Annahmen und Einschränkungen

3.4 Organisationsstruktur

Vorname	Name	E-Mail	Verantwortlich für
Andreas	Stalder	astalder@hsr.ch	-
David	Meister	dmeister@hsr.ch	-

Tabelle 2: **Teammitglieder**

3.5 Externe Schnittstellen

Das Projekt wird von Beat Stettler und Urs Baumann betreut und benotet.

4 Management Abläufe

4.1 Zeitbudget

Der Projektstart ist am Montag, dem 22. September 2016.

Die Projektdauer beträgt 14 Wochen, und das Projektende ist am Freitag, dem 23. Dezember 2016.

Während diesen 14 Wochen sind 240 Arbeitsstunden pro Projektmitglied eingeplant. Das entspricht pro Mitglied eine Arbeitszeit von ca. 18 Stunden pro Woche. Dies ergibt einen totalen Aufwand von ca. 480 Stunden.

Die wöchentliche Arbeitszeit von 18 Stunden kann bei Verzug oder bei unerwarteten Problemen auf maximal 24 Stunden erhöht werden.

Es sind gegenwärtig keine Absenzen während dieser Zeit geplant.

4.2 Zeitliche Planung

Die Zeitplanung und die Verwaltung der Arbeitspakete erfolgt in Redmine. Diese wird während dem Projekt laufend aktualisiert. Die in Redmine erzeugten Tickets dienen sogleich 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-Diagramm via URL: Die Projektmitglieder tragen jeweils die investierte Zeit am Abend in das zugewiesene Ticket ein.

4.2.1 Phasen

Das Projekt wird in fünf Phasen unterteilt: Initialisierung, Analyse, Design, Realisierung und Abschluss.



4.2.2 Meilensteine

Das Projekt beinhaltet insgesamt fünf Meilensteine.

Meilenstein	Beschreibung	Datum
MS1	Anforderungen und Scope definiert	13.10.16
MS2	Architektur und Design beschrieben	27.10.16
MS3	Betaversion fertiggestellt	17.11.16
MS4	Software und Dokumentationen fertiggestellt	08.12.16
MS5	Arbeitsabgabe	23.12.16

Tabelle 3: **Projekt Meilensteine**

4.2.3 Iterationen

Die Dauer eines Iterationszyklus beträgt jeweils eine Woche.

Iteration	Inhalt	Start	Ende
Initialisierung	Projektstart und Kickoff-Meeting	15.09.2016	22.09.2016
Analyse 1	Projektplanung	23.09.2016	30.09.2016
Analyse 2	-	01.10.2016	06.10.2016
Analyse 3	-	07.10.2016	13.10.2016
Design 1	-	14.10.2016	20.10.2016
Design 2	-	21.10.2016	27.10.2016
Realisierung 1	-	28.10.2016	03.11.2016
Realisierung 2	-	04.11.2016	10.11.2016
Realisierung 3	-	11.11.2016	17.11.2016
Realisierung 4	-	18.11.2016	24.11.2016
Realisierung 5	-	25.11.2016	01.12.2016
Realisierung 6	-	02.12.2016	08.12.2016
Abschluss 1	-	09.12.2016	15.12.2016
Abschluss 2	-	16.12.2016	23.12.2016

Tabelle 4: **Projekt Iterationen**

4.2.4 Arbeitspakete (Tickets)

Name	Inhalt	Iteration	Wer	Soll	Ist
Projektstart					
Kickoff-Meeting	Allgemeine Besprechungen zum Projektstart	Initialisierung	Alle	1	0.75

Tabelle 5: Arbeitspakete

4.3 Teammeeting

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

4.3.1 Meeting mit Betreuern

Die Meetings mit den Betreuern finden jeden Donnerstag um 13:30 Uhr statt. Die Meetings werden mit den Betreuern Beat Stettler und Urs Baumann in ihrem Büro durchgeführt. Die Dauer eines Meetings ist unterschiedlich und kann stark variieren.

5 Risikomanagement

5.1 Risiken

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

5.2 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 anderen Lösung (z.B. Einschränkung von Programmfeatures, etc.) gesucht werden.

6 Qualitätsmassnahmen

Massnamen	Zeitraum	Ziel der Massnahme
git verwenden	immer	Versionierung und Verhinderung von Datenchaos
Redmine verwenden	immer	Einhaltung von Vorgehen und Zeitplan
Teamsitzung	wöchentlich	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
Continuous Integration	immer	Steigerung der Softwarequalität

Tabelle 6: **Qualitätsmassnahmen**

6.1 Dokumentation

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

6.2 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 erfolgt über die Url: - Um den Zugriff für Betreuungspersonen zu ermöglichen wurde ein Gastbenutzer eingerichtet.

Logindaten Redmine Gastbenutzer:

Login: -

Password: -

6.3 Entwicklung

6.3.1 Versionierung

Wie die Dokumentation wird auch der Sourcecode mit git versioniert und auf GitHub abgelegt. Es wird darauf geachtet, möglichst häufig auf den Stamm zu commiten.

6.3.2 Test Driven Development

Es hat sich herausgestellt, dass die Software Qualität mit Test Driven Development (TDD) am besten ist. Beim TDD wird vor der Implementation der dazugehörige Unit Test entwickelt. Wo überall möglich, wird mittels TDD entwickelt.

6.3.3 Code Metriken

6.3.4 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.

6.3.5 Continuous Integration