Projektplan

Studienarbeit FS-2020

26. Februar 2020

Autoren:

Mike Schmid
mike.schmid@hsr.ch

Janik Schlatter
janik.schlatter@hsr.ch

Supervisor:

Prof. Stettler Beat
beat.stettler@hsr.ch

Zweck

Dieses Dokument beschreibt den Projektplan und liefert eine Übersicht über das Projekt Network Unit Testing System, dessen Planung und Organisation, sowie über weitere Bereiche des Projektaufbaus. Der Projektplan dient als Grundlage und Referenz für nachfolgende Projektdokumente

Änderungsgeschichte

Datum	Version	Änderung	Autor
20.02.2018	1.0	Initial Setup	Janik Schlatter

Inhaltsverzeichnis

1 Einführung

1.1 Sprache

Die allgemeine Projektsprache (Dokumentation, Use Cases, etc.) wird in Deutscher Schriftsprache verfasst. Der Code, das GitHub-Repository und die Versionskontrolle wird in Englischer Sprache geschrieben.

1.2 Referenzen

Alle Dokumente werden auf dem GitHub-Repository abgelegt und verwaltet.

Git Repository	`https://github.com/EkoGuandor229/Network-Unit-Testing.git`
Vorarbeit NUTS	`https://github.com/HSRNetwork/Nuts.git`

1.3 Vorarbeit NUTS

Die Studienarbeit aus dem Jahr 2016 hat ein Programm erarbeitet, die mit SaltStack, einer Lösung für die Automatisierung von Projekten, das Testen von Netzwerkumgebungen mittels Python ermöglicht. Dabei wurden umfangreiche Tests in der Serialisierungssprache YAML spezifiziert und umgesetzt. Die Schwierigkeit lag vor allem darin, dass nicht jedes Netzwerkgerät dieselben Funktionen für spezifische Befehle bietet, da Hersteller Unterschiedliche Befehle für ihre Geräte verwenden. Darauf aufbauend wird in dieser Arbeit die Testdefinition weiter verwendet und nach Bedarf ergänzt oder ausgebaut.

2 Projektübersicht

2.1 Projektübersicht

2.2 Zweck und Ziel

Das Testen von Netzwerkkonfigurationen findet auch heute noch hauptsächlic mit handgeschriebenen CLI-Befehlen oder kleinen Skripten statt. Wenn der Netzwerktechniker einen Test vergisst, oder die Formulierung nicht stimmt, kann es vorkommen, dass im Netzwerk Fehler auftreten, deren Ursprung schwierig zu ermitteln ist und eine komplette Repetition der (handgeschriebenen) Tests erfordert. Ein Programm, welches wie in der Softwareentwicklung vordefinierte und automatisch durchgeführte Tests, sogenannte Unit-Tests, ermöglicht, könnte diese Probleme stark verringern. Dabei können zwei grobe Arbeitsvorgänge beschrieben werden. Im ersten schreibt ein Netzwerktechniker Tests, die ein bestehendes Netzwerk möglichst genau abbilden/beschreiben sollen. Die tests lassen sich jederzeit durchführen und testen den Zustand und die Konfiguration des Netzwerks. Falls nun ein Fehler auftritt, können die Tests automatisiert durchgeführt werden und dann, vorausgesetzt sie sind vollständig, sollte der Report aufzeigen, was genau schiefgegangen ist und wo der Fehler liegt. Der Zweite Arbeitsvorgang entspricht dem in der Softwareentwicklung gängigen Test-Driven-Developmet (TDD). Beim TDD werden Tests geschrieben, bevor das System verändert wird oder bevor man neuen Code schreibt. Auf ein Netzwerk abstrahiert könnte beispielsweise ein Administrator, der eine Änderung am Netzwerk vornehmen will, zuerst die Tests schreiben, welche die Änderung testen sollen. Danach werden die Konfigurationen verändert und die Tests durchgeführt. Falls die Tests nun fehlschlagen, kann man die Konfiguration anpassen oder sogar auf einen früheren Zustand zurücksetzen. Beide Arbeitsvorgänge erleichtern die Fehlersuche und erhöhen die Stabilität des Netzwerks.

2.3 Lieferumfang

Im Rahmen der Studienarbeit wird folgendes erstellt:

- Eine Überarbeitung der Testdefinionssprache (TDS), die in der Vorarbeit ausgearbeitet wurde.
- Python-Software, die die TDS umsetzt und auf Netzwerkinfrastrukturen Tests durchführen kann.

2.4 Annahmen und Einschränkungen

3 Projektorganisation

3.1 Projektmitglieder

Name	Email
Janik Schlatter	jschlatt@hsr.ch
Mike Schmid	mschmid@hsr.ch

3.2 Externe Schnittstellen

Name	Email	Zuständikeit
Beat Stettler	beat.stettler@hsr.ch	Betreuer
Urs Baumann	urs.baumgartner@hsr.ch	Betreuer

4 Management Abläufe

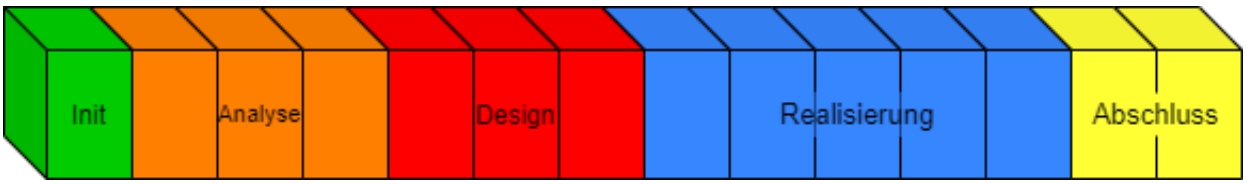
4.1 Kostenvoranschlag

Das Projekt wurde am 20.02.2020 gestartet und wird voraussichtlich am 28.05.2020 enden. Das heisst, es stehen 15 Wochen während dem Semester zur Verfügung. Jedes Projektmitglied arbeitet insgesamt 240 Stunden an dem Projekt, sprich 16 Stunden pro Woche pro Projektmitglied. Da der Dienstag 14.04.2020 und der Donnerstag 21.05.2020 jeweils ein Unterrichtsfreier Tag ist, werden die 16 Stunden pro Projektmitglied auf jeweils vier Samstage im Projekt verteilt aufgeteilt. Diese Samstage dienen dem Ziel, die Dokumentation nachzutragen und die Risiken, wie sie im Kapitel 5 beschrieben sind, zu minimieren. Dazu gehören Bugfixing, Recherchen und Aufarbeiten von Themen, die ungenügend verstanden sind und Refactoring des Code.

Projektdauer	15 Wochen
Anzahl Projektmitglieder	2
Arbeitsstunden pro Woche und Person	16
Arbeitsstunden insgesamt	480
Projektstart	20.02.2020
Projektende	28.05.2020

4.2 Zeitliche Planung

Die 14 Wochen des Projekts werden in fünf Phasen unterteilt: Initialisierung, Analyse, Design, Realisierung und Abschluss.



4.3 Phasen/Iterationen

Phasen

Wir halten uns an die folgenden 5 Phasen:

Farbe*	Bezeichnung	Zeitrahmen
Grün	Initialisierung	1 Woche
Orange	Analyse	3 Wochen
Rot	Design	3 Wochen
Blau	Realisierung	5 Wochen
Gelb	Abschluss	2 Wochen

Iterationen

Die Iterationen werden wöchentlich durchgeführt. Da wir auch ein Mal in der Woche das Meeting haben passt das gut aufeinander.

4.4 Meilensteine

Nr	Bezeichnung	Termin	Beschreibung
M1	Projektplan	So 01.03.2020	Grundentwurf der Requirements, Risikoanalyse & -management, Projektorganisation, Managementabläufe, Infrastrukturentwurf, Qualitätsmassnahmen Grundentwurf.
M2	Requirements	So 15.03.2020	Ausgearbeitete Requirements, Nichtfunktionale Anforderung, Zu verwendende Tools und Schnittstellen beschrieben.
M3	Prototyp	So 05.04.2020	Architektur festgeleg, Schnittstellen angelegt, Architekturdokumentation, Testprozeduren eingerichtet und UnitTests erstellt, Erster lauffähiger Prototyp.
M4	Feature Freeze	Do 07.05.2020	Hauptfunktionalität der Software implementiert, Bugs sind bekannt und Dokumentiert, Codedokumentation zu 60% fertiggestellt.
M5	Codefreeze & Codeabgabe	Di 19.05.2020	Bugfixes erstellt, Tests sind alle erfolgreich, Codedokumentation zu 60% fertiggestellt.
M6	Projektabgabe	Do 28.05.2020	Dokumentation fertiggestellt und abgegeben.

4.5 Besprechungen/Protokolle

Es wurden zwei Termine vereinbart, an welchen sich die Projektmitglieder treffen. Bei beiden Terminen stehen jeweils mindestens 6 Lektionen zur Verfügung.

Nr	Wann	Beschreibung
1	Dienstag 10:00 - 17:00	Gemeinsame Arbeit der Projektmitglieder
2	Donnerstag 08:00 - 17:00	Gemeinsame Arbeit der Projektmitglieder
3	Donnerstag 14:00 - 15:00	Besprechung mit Projektbetreuern
4	Samstag 14.03.2020	Dokumentation und Risikoreduktion
5	Samstag 28.03.2020	Dokumentation und Risikoreduktion
6	Samstag 11.04.2020	Dokumentation und Risikoreduktion
7	Samstag 25.04.2020	Dokumentation und Risikoreduktion
8	Samstag 09.05.2020	Dokumentation und Risikoreduktion
9	Samstag 23.05.2020	Dokumentation und Risikoreduktion

5 Risikomanagement

5.1 Risiken

Eine Riskoanalyse mit gewichtetem Schaden und Informationen zur Vorbeugung ist auf der Ablage zu finden (siehe Dokument TechnischeRisiken.xlsx)

5.2 Umgang mit Risiken

Um Probleme gerade während der Init/Analyse Phase möglichst früh zu erkennen, arbeiten wir wöchentlich zwei Tage nebeneinander, um uns über mögliche Probleme auszutauschen. Desweiteren suchen wir auch den Kontakt zum Betreuer sobald Unklarheiten im Team herrschen. Einmal alle zwei Wochen treffen wir uns am Samstag für drei bis vier Stunden, um an der Dokumentation und and der Reduktion aufgetretener Risiken zu arbeiten. Die genauen Termine werden im Kapitel 4.5 beschrieben.

6 Infrastruktur

Alle Arbeiten zum Projekt werden von den Projektmitgliedern auf Ihrem jeweiligen Laptop verrichtet. Alternativ stehen in den Studienarbeitszimmern Computer zur Verfügung, mit denen im falle eines Geräteausfalls weitergearbeitet werden kann. Für das Testen der Software wird vom Institut für Networked Solutions eine Routerinfrastruktur zur Verfügung gestellt. Die genauen Ausmasse der Infrastruktur sind zum Zeitpunkt der Erstellung des Projektplans noch nicht spezifiziert.

6.1 Übersicht der Tools

Für die Umsetzung des Projektes werden folgende Tools verwendet:

Bezeichung	Beschreibung
Git	Versionsverwaltung
PyCharm	Entwicklungsumgebung
Visual Studio Code	Editor für die Dokumentation

7 Qualitätsmassnahmen

7.1 Allgemein

7.2 Testing

Für das Unti-Testing wird das Python-Modul PyTest verwendet, ein Framework, welches das einfache Testen von Pythoncode erlaubt.

7.3 Besprechungen

7.4 Versionskontrolle

Sämtliche Dokumente werden in einem Git Repository abgelegt. Damit wird, dank der Versionskontrolle, jede Änderung nachvollziehbar und es können auf sämtlichen alten Versionen zugegriffen werden.

7.5 Dokumente

7.6 Code-Qualität

Um die Codequalität zu gewährleisten, werden folgende Massnahmen ergriffen:

- Unittests, um Fehler im Code zu verhindern oder früh zu finden.
- In der Versionsverwaltung wird mit Feature-Branches gearbeitet, damit Änderungen nicht zu Merge-Konflikten führen.
- Merges in den Development-Branch dürfen nur nach einem erfolgreichen Pull-Request durchgeführt werden.
- Pull-Requests werden vom jewils anderen Teammitglied überprüft und allfällige Probleme müssen vor der Annahme korrigiert werden.
- Im Git wird ein CI/CD (Continous Integration/Continous Delivery) aufgesetzt, das bei Änderungen im Development-Branch Tests durchführt.
- Getestet wird die einhaltung der Coderichtlinien, Vorherig definierte Tests und das erfolgreiche durchlaufen des Build-Prozesses.
- Bekannte Bugs werden mittels Git Issues erfasst und verwaltet. Das Abarbeiten dieser Issues ist teil des Projekts und beim Meilenstein M5:Codefreeze & -abgabe sollten sich keine Issues mehr auf dem Repository befinden