Network Unit Testing System

Studienarbeit FS-2020

22. Mai 2020

Autoren:

Mike Schmid
mike.schmid@hsr.ch

Janik Schlatter
janik.schlatter@hsr.ch

Supervisors:

Prof. Stettler Beat
beat.stettler@hsr.ch

Baumann Urs
urs.baumann@hsr.ch

Aufgabenstellung

Änderungen an Netzwerkumgebungen werden in der Praxis auch heute noch durch Kommandozeilenbefehle oder Skripte getestet. Diese Tests beinhalten oft einfache Befehle wie 'ping' oder 'traceroute'. Im Vergleich dazu werden Softwareprojekte durch automatisierte Tests, welche regelmässig ausgeführt werden, getestet. Sogenannte Unit Tests werden vor und nach einer Änderung durchgeführt, um zu Testen, ob sich ein Programm weiterhin innerhalb der geforderten Betriebsparameter verhält. Somit können Fehler schnell gefunden und behoben werden und die Robustheit der Software wird erhöht. Ein vergleichbarer Arbeitsablauf soll auch für den Netzwerkbereich ermöglicht werden.

Eine frühere Studienarbeit hat sich mit der Entwicklung einer Beschreibungssprache befasst, mit derer solche Tests möglich wären. Die Vorarbeit wurde dabei so entwickelt, dass das Programm mit dem Automationsframework SaltStack ausgeführt wurde. Diese Arbeit soll ein Programm entwickeln, welches selbstständig und unabhängig von anderen Programmen arbeiten kann.

Die Studierenden erhalten die Aufgabe, ein Python-Programm zu entwickeln, welches automatisierte Tests auf ein Netzwerk durchführen kann. Das Programm soll gemäss einer Testdefinition selbstständig die auszuführenden Tests erstellen, durchführen und die Testresultate mit einem Erwartungswert vergleichen. Die Auswertung der Tests soll direkt bei der Ausführung auf der Konsole angezeigt werden und zusätzlich in einem Testreport für die spätere Ansicht gespeichert werden. Die Programmausführung kann manuell oder automatisch über einen Deployment-Prozess gestartet werden.

Prof.Beat Stettler

Urs Baumann

Abstract

Das testen von Netzwerkkonfigurationen findet auch heute noch hauptsächlich mit handgeschriebenen CLI-Befehlen oder kleinen Skripten statt. Wenn der Netzwerktechniker einen Fehler bei der Konfiguration macht, oder etwas vergisst, 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-Development (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.

Aus dieser Arbeit ist das Programm "Nuts2.0" hervorgegangen, die vordefinierte Netzwerktests mit dem Automatisierungsframework Nornir durchführt und die Ergebnisse ausgewertet darstellt. Nornir ermöglicht es, dass unterschiedliche Geräte von verschiedenen Herstellern über mehrere Kommunikationskanäle angesprochen werden können und die Testresultate in einer einheitlichen Formatierung zurückgegeben werden.

Management Summary

Ausgangslage

Fehler in Teilbereichen von Netzwerksystemen können dazu führen, dass das ganze System nicht mehr funktioniert. Aus diesem Grund ist es essentiell, dass selbst kleine Änderungen an Netzwerken getestet werden können. Diese Tests werden meistens von Hand oder durch Skripte durchgeführt. Ein Tool, welches das automatisierte Testen von Netzwerksystemen ermöglicht, kann dabei helfen, Fehler zu erkennen, bevor sie zu einem Problem werden.

Vorgehen, Technologien

Zu Beginn wurde eine Domänenanalyse durchgeführt, um die Akteure und Bestandteile einer Netzwerkumgebung zu bestimmen und die zu entwickelnden Netzwerktests zu evaluieren. Darauf aufbauend wurden die funktionalen und nichtfunktionalen Anforderungen an die Software spezifiziert. Auf dieser Basis wurde die Softwarearchitekur ausgearbeitet und mit der Entwicklung begonnen.

Das Programm wurde in der Programmiersprache Python geschrieben und beinhaltet das Modul "Nornir", ein Framework, welches automatisierte Tasks auf Netzwerksysteme, wie z.B. Konfiguration oder Tests, ermöglicht.

Ergebnisse

Aus dieser Studienarbeit ist ein Python-Programm entstanden, welches Netzwerktests, die in einer Definitionssprache spezifiziert werden, gegen ein Netzwerk durchführt und die Ergebnisse selbstständig auswertet und dem Benutzer anzeigt. Nornir erlaubt dabei, eine Vielzahl von Netzwerkgräten anzusprechen, welche über herkömmliche Methoden wie SSH umfänglicher zu testen wären.

Die Software lässt sich ohne Installation auf jedem Gerät ausführen, welches Python-Code ausführen kann, unabhängig vom Betriebssystem. Geräte, auf denen Python nicht installiert ist, müssen dies zuerst Installieren, können das Programm danach aber ohne weiteres ausführen.

Dadurch, dass das Programm reiner Pyhton Code ist, lässt es sich einfach in ein bestehendes Tool für die kontinuierliche Integration einbinden. Die Testdefinitionen lassen sich über ein Versionsverwaltungstool zentralisieren, so dass mehrere Netzwerkleute gleichzeitig Tests für eine Umgebung entwickeln können.

Inhaltsverzeichnis

I. Technischer Bericht

1 Einleitung

1.1 Problemstellung

Netzwerke bestehen aus dutzenden bis tausenden Komponenten. Jede dieser Komponente hat eine eigene Konfiguration und Aufgabe. In den letzten Jahren ist es durch die softwaregesteuerte Konfiguration von Netzwerkgeräten einfacher geworden, die Komponenten für ihre Aufgabe einzustellen. Trotzdem werden die Überprüfungen der Konfiguration auch heute noch manuell vorgenommen. Dies kann dazu führen, dass wegen menschlicher Fehler ein Fehlverhalten eines der Netzwerkkomponente dazu führt, dass das gesammte Netzwerk gestört wird. Weiterhin wird die Überprüfung noch komplizierter, da neben der statischen Konfiguration sich das Netzwerk zur Laufzeit dynamisch anpasst, um die Performanz des Systems zu optimieren und Fehler sowie Ausfälle zu korrigieren. Dieses Verhalten wird über verschiedene Netzwerkprotokolle gesteuert, z.B. OSPF oder BGP.

In der Softwareentwicklung werden schon seit Ende der 80er-Jahre Komponententests, sogenannte Unit-Tests durchgeführt, um einzelne Komponenten (Units) automatisiert zu Testen. Dabei wird ein Computerprogramm ausgeführt, welches mit verschiedenen Eingabeparametern überprüft, ob die Ausgabe des zu testenden Programms den erwarteten Ergebnissen entspricht.

Dabei ist es möglich die Tests vor und nach einer geplanten Änderung durchzuführen, um zu überprüfen, ob die Software innerhalb der definierten Funktionsparameter operiert. Tests sollen dafür so geschrieben werden, dass möglichst jede Situation mit den Eingabeparametern abgebildet wird. Unittests sollen regelmässig durchgeführt werden, damit Fehler früh gefunden werden und sich nicht auf das gesammte System auswirken können.

1.2 Aufgabenstellung

Ziel dieser Arbeit ist, ein Programm zu entwickeln, mit dem sich Netzwerktests mit der gleichen Arbeitsweise durchführen lassen, wie Unittests in der Softwareentwicklung durchgeführt werden. Dabei müssen folgende Anforderungen an die Tests erfüllt sein:

- Tests müssen planbar sein. Es soll ein Testplan existieren.
- Tests müssen systematisch spezifiziert werden. Es existieren Test-Spezifikationen.
- Testresultate werden Dokumentiert.
- Tests sollen, wo möglich, automatisiert durchgeführt werden.
- Testergebnisse müssen reproduzierbar und nachvollziehbar sein.

Es soll evaluiert werden, welche bereits verfügbaren Tools sich für ein solches Testprogramm eignen. Die Umsetzung soll diese Tools einbinden. Eine Wichtige Anforderung an das zu entwickelnde System ist, dass sich weitere Netzwerktests möglichst einfach hinzufügen lassen, ohne dass dafür der gesamte Code geändert werden muss.

1.3 Herausforderungen

Eine der grössten Herausforderungen an ein automatisiertes Testsystem sind die verschiedenen Protokolle und die Unterschiede der Standards von diversen Herstellern.

Ohne Kenntnisse, welche Protokolle auf einem Netzwerkgerät konfiguriert sind, können diese Netzwerkgeräte nicht effizient getestet werden, da die Netzwerkprotokolle das Verhalten der Geräte zur Laufzeit beeinflussen. Deshalb müssen für die Testdurchführung im vornherein die Konfigurationen und verwendeten Protokolle der Netzwerkgeräte bekannt sein und ein Testsystem muss mit diesen interagieren können.

Unterschiedliche Hersteller haben verschiedene Kommandozeilenbefehle (CLI-Commands) für ihre Geräte, welche für die Konfiguration und Abfrage der Konfiguration verwendet werden. Ausserdem kann es vorkommen, dass ein Hersteller mit der Einführung einer neuen Version des Gerätebetriebssystems neue CLI-Commands einführt oder alte Befehle ändert. Dies setzt eine enorme flexibilität für ein Testprogramm voraus, welches ein beliebiges Netzwerk mit unterschiedlichen Geräten von verschiedenen Herstellern testen soll.

1.4 Vorarbeit

TODO

2 Ergebnisse

Die Studienarbeit hat ein Python-Programm entwickelt, welches basierend auf einer Testdefinition im YAML-Format automatische Tests gegen ein Netzwerk ausführen kann.

Kern der Software ist das Nornir-Modul für Python, ein Framework welches in Python geschrieben ist und die Automation von Netzwerktätigkeiten ermötlicht. Nornir erlaubt dem Anwender, automatisiert Konfigurationsänderungen oder -abfragen an ein Netzwerk zu senden.

Dabei stehen verschiedene Netzwerkschnittstellen zur Verfügung:

Napalm Abkürzung für "Network Automation and Programmability Abstraction Layer with Multivendor support". Napalm ist eine Pyhton Library welche verschiedene Funktionen anbietet, mit denen man mit Netzwerkgeräten über eine einheitliche Schnittstelle kommunizieren kann.

Paramiko Paramiko ist eine Python-Implementation des SSHv2 Protokolls für die sichere Kommunikation zwischen verschiedenen Endgeräten.

Netmiko Netmiko ist eine Library, welche die Paramiko SSH verbindung vereinfacht. Das Ziel von Netmiko ist, für verschiedene Herstellergeräte eine einheitliche Schnittstelle zu bieten und die Kommunikation zwischen Endgeräten und Server zu vereinfachen.

Netconf Das Network Configuration Protocoll ist ein Protokoll für das Netzwerk Management. Netconf wurde als RFC 6241 publiziert und bietet Mechanismen für die Installation, Manipulation und das Löschen von Konfigurationen auf Netzwerkgeräten.

2.1 Architekturüberblick

Die erarbeitete Software baut sich aus fünf Kernbereichen und einer Utility-Gruppe zusammen.

Inventory-Management Das Inventory umfasst die Klassen, welche für die Verwaltung des Inventars benötigt werden. Dazu gehören die Verwaltung der Devices und deren Verbindungen sowie die Verwaltung der Testdefinitionen.

Devices und Deviceconnections beschreiben die Netzwerkgeräte und die Verbindungen zwischen den Geräten. Diese Informationen werden vom Testprogramm benötigt, um die konkreten Tests für das jeweilige Gerät evaluieren zu können, da z.B. ein traceroute-Befehl mit einem Netmiko Befehl ausgeführt werden kann und mit einem Napalm-Befehl nicht ausführbar ist. Ausserdem lassen sich bestimmte Napalm-Befehle auf spezifischen Betriebssystemen (z.B. Cisco-IOS) ausführen, funktionieren aber auf anderen Systemen (Cisco NXOS) nicht.

Die Testdefinitionen beschreiben, welche Tests auf welchen Geräten ausgeführt werden sollen. Sie beinhalten den Testnamen, den Testcommand, das zu erwartende Ergebniss und weitere Parameter, wie beispielsweise das Zielgerät für einen Ping-Test.

Resources Im Resourcen-Ordner sind die YAML-Dokumente gespeichert, welche die Devices, Deviceconnections und die Testdefinitionen beschreiben. Ausserdem werden die die Testergebnisse im Unterordner TestResults im .txt-Format abgelegt.

Das YAML Format wurde ausgewählt, da die Beschreibungssprache für Netzwerkleute bereits bekannt ist und sich einfach interpretieren und schreiben lässt.

Die Testdefinitionen können alle in einem Dokument oder in mehreren Dokumenten verfasst werden, um bei Bedarf die Definitionen nach Art der Tests oder nach Bereichen im Netzwerk zu sortieren.

Connection-Handling Das Connection-Handling befasst sich mit der Logik, welche Netzwerkschnittstelle für welches Device-Betriebssystem und -version verwendet werden soll. Nach Möglichkeit wird dabei der Napalm-Treiber von Nornir verwendet und der Netmiko-Treiber als Fallback-Lösung implementiert.

Testcreation Die Testerstellung befasst sich mit der Instanzierung der konkreten Netzwerktests. Dafür wird eine Kombination der beiden Software Patterns Strategy Pattern und Factory Pattern angewendet, um die Tests zur Laufzeit gemäss der Testdefinitionen mit den Korrekten Netzwerkschnittstellen zu instanzieren. Jeder Netzwerktest beinhaltet dabei die Logik, welche für die Ausführung, Evaluation und Reporting benötigt werden. Somit müssen in den Klassen, die diese Funktionalitäten ausführen lediglich die Methoden der jeweiligen Tests aufgerufen werden.

Das Strategy Pattern ist ein Verhaltenspattern und spezifiziert eine Gruppe von Strategien, in diesem Fall Netzwerktests, die basierend auf einem Kontext implementiert werden. Die Anwendung dieses Patterns erlaubt das Austauschen einer Strategie mit einer anderen zur Laufzeit, ohne dass die Programmlogik desshalb verändert wird. Dadurch erhöht sich die Wiederverwendbarkeit und Flexibilität der Implementierung.

Das Factory Pattern ist eines der meistverwendeten Patterns in der Softwareentwicklung und erlaubt die Trennung der Instanzierungslogik von dem Verhalten einer Klasse. Dabei wird eine Factory-Klasse implementiert, welche eine Methode beinhaltet, die die Konkreten Klassen instanziert, welche vom rest des Programms verwendet wird. Der grösste Vorteil der Verwendung dieses Patterns ist die einfachere Erweiterbarkeit des Programms, da nur die Instanzierungslogik in der Factory angepasst werden muss, wenn eine neue Testklasse implementiert werden soll.

Testhandling Das Testhandling befasst sich mit der Erstellung, Ausführung, Evaluation und dem Reporting der einzelnen Netzwerktests.

Der Netzwerk-Test-Builder erzeugt die Netzwerktest, indem die Methoden der oben genannten Testerstellung aufgerufen werden.

Mit der Netzwerk-Test-Order können die Tests bei bedarf über eine Grafische Benutzeroberfläche (GUI) ausgewählt und die Ausführungsreihenfolge definiert werden. Wird das Programm aus der Kommandozeile ausgeführt, kann die Auswahl mit dem GUI übersprungen werden, damit sich das Programm auch automatisch mit einer Orchestrierungssoftware ausführen lässt. In dem Fall werden die Tests in der Reihenfolge ausgeführt, in welcher sie in der Testdefinitionen spezifiziert sind.

Der Netzwerk-Test-Runner führt die Netzwerktest, welche in der Testdefinition spezifiziert sind, gegen das Netzwerk aus.

Der Evaluator vergleicht die Rückgabewerte der ausgeführten Tests mit dem Erwartungswert welcher in der Testdefinition spezifiziert ist und entscheidet aufgrund der Evaluation, ob ein Test bestanden oder fehlgeschschlagen ist.

Der Reporter gibt die Ergebnisse der Testdurchführung auf der Konsole aus und speichert sie in einem Resultat-Dokument. Die Testergebnisse werden dabei nach bestandenen und fehlgeschschlagen Tests sortiert. Bei den bestandenen Tests wird nur der Testname ausgegeben. Ist ein Test fehlgeschschlagen,

wird der Testname, das erwartete Ergebnis und das tatsächliche Ergebnis ausgegeben, damit direkt ein Vergleich vorgenommen werden kann.

Utilities Zu den Utilities gehören der Filehandler, der Progressbar-Handler und der UI-Handler.

Der Filehandler liest und schreibt informationen in den YAML und TXT Dokumenten aus, die während der Ausführung der Tests benötigt werden.

Der Progressbar-Handler und UI-Handler erzeugen das Userinterface in der Kommandozeile während der Programmdurchführung.

3 Schlussfolgerungen

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

4 Glossar

Begriffserklärung

TODO

Abbildungsverzeichnis

*

Tabellenverzeichnis

*

Literaturverzeichnis

TODO

II. Anhang

1 Projektplanung

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

2 Anforderungen

Im folgenden Abschnitt werden die Anforderungen an ein Netzwerk-Test-System formuliert. Es werden die Kern-Akteuere identifiziert und deren Funktion und Abhängigkeiten und Anforderungen formuliert, um auf dieser Basis die Software zu entwerfen.

2.1 Akteure in einem Netzwerksystem

In der Praxis gibt es für die verschiedenen Akteure in einem Netzwerksystem unterschiedliche Bezeichnungen. Beispielsweise ist oft nicht klar, was der Unterschied zwischen einem Netzwerk-Architekten und einem Netzwerk-Engineer ist und welche Verantwortungen diese nun genau haben. Wir haben eine eigene Unterscheidung der Akteure formuliert und diese in den kommenden Sektionen dokumentiert, um eine einheitliche Basis für die Leser zu schaffen.

Netzwerk-Architekt

Ein Netzwerk-Architekt plant und erstellt Kommunikationsnetzwerke. Im Zuge dieser Arbeit wurde zwischen dem Architekten als verantwortlichen Senior-Network-Engineer und einem Network Engineer (Junior oder Senior) als operativen Mitarbeiter unterschieden. Der Architekt nimmt dabei eher die Rolle des Managers oder Teamleiters ein. Er führt dabei normalerweise keine Konfigurationen am Netzwerk durch.

Netzwerk-Engineer

Ein Netzwerk-Engineer ist für die Installation und Instandhaltung eines Netzwerks zuständig. Er ist dem Netzwerk-Architekten unterstellt und setzt mit Ihm zusammen die geplanten Arbeiten um.

Netzwerk-Administrator

er Netzwerk Administrator hat üblicherweise eine abgeschlossene Berufslehre in der Informatik und arbeitet zusammen mit dem Netzwerk-Engineer am Netzwerk. Es wird davon ausgegangen, dass ein Netzwerk Administrator keine bis wenige Programmierkenntnisse hat. Ein Netzwerk Administrator hat, je nach Grösse des Netzwerks, nur Kenntnisse über einen Teil der Netzwerkumgebung. Er führt dabei ihm vom Architekten oder Engineer vorgegebene Arbeiten aus und muss dazu nicht den vollen Überblick über das Netzwerk und die darin verwendeten Technologien haben.

Netzwerk-User

Benutzer der Netzwerkumgebung. User können das Netzwerk verwenden, aber nicht dessen Konfigurationen anpassen.

Netzwerk-Gerät

Ein Netzwerkgerät kann aus Hardware wie Switch, Router oder Server bestehen oder Virtuell als Software implementiert sein. Im Zuge der Arbeit werden Netzwerkgeräte auch als Netzwerk-Devices oder einfach Device bezeichnet. Typischerweise haben Devices eine Statische Konfiguration und einen dynamischen Zustand zur Laufzeit. In den kommenden Kapiteln wird genauer auf Netzwerkgeräte eingegangen.

Netzwerk-Verbindung

Die Netzwerkverbindung ist der Kommunikationskanal zwischen den einzelnen Netzwerkgeräten. Sie kann in physischer Form als Kabel, oder mit kabellosen Mitteln z.B. Funk umgesetzt sein. Die Wahl des Übertragungsmediums hat grossen Einfluss über die verfügbare Bandbreite und mögliche Störfaktoren.

Repository/Inventar

Im Inventar werden die Unterschiedlichen Devices mit den für den Betrieb wichtigsten Parametern gespeichert. Das Inventar kann in digitaler Form als Repository, als File auf einem Ordner/Computer, oder analogin einem Dokumentenorder abgelegt sein. Das Inventar wird benötigt, um die aktuellen Konfigurationen, die physische Position des Geräts oder sonstige für den Betrieb relevanten Informationen zu dokumentieren.

2.2 Akteure in der zu entwickelnden Software

Testprogramm

Das Testprogramm ist der Kern des zu entwickelnden Systems dieser Arbeit. Es Interagiert mit den anderen Akteuren und hat, vom Akteur und Kontext abhängig, unterschiedliche Anforderungen. Es soll so aufgebaut sein, dass ein Benutzer der Software diese mit möglichst geringem Aufwand bedienen kann.

Testdefinitionssprache

Wird im Rahmen dieser Arbeit auch als Testbeschreibungssprache oder Definitionssprache bezeichnet. Eine Testdefinition beschreibt die einzelnen Testfälle, die von einem System durchgeführt werden sollen. Die Definitionssprache soll dabei in einem Format gehalten werden, das von allen Benutzern der Software verstanden wird und von diesen erweitert werden kann.

Testreport

Ein Testreport soll, möglichst einfach und genau, die Ergebnisse eines Netzwerktests aufzeigen. Fehlgeschlagene Tests sollen dabei möglichst einfach und schnell zu erkennen sein und alle Informationen beinhalten, die ein Betrachter benötigt, um den Fehler im System zu lokalisieren und beheben. Ausserdem muss mindestens noch ein Zeitstempel vorhanden sein, um die Historie vergangener Netzwerktests nachvolziehen zu können. Testreporte können auf dem System, welches die Tests ausführt, oder in einem zentralen Repository abgelegt werden, damit mehrere Mitglieder eines Netzwerkteams gleichzeitig darauf zugreifen können.

Kommunikationskanal

Der Kommunikationskanal, nicht zu verwechseln mit der Netzwerkverbindung zwischen zwei Netzwerkgeräten, verbindet ein zu testendes Netzwerk mit dem Testprogramm. Möglichkeiten für einen Kanal sind beispielsweise das SSH (secure shell) Protokoll oder der Restconf Standard. Dies kann über eine Kabelverbindung oder Kabellos geschehen. Die Wahl des Kommunikationskanals beeinflusst dabei, in welcher Form Netzwerktests durchgeführt werden können und in welchem Format die Ergebnisse zurückgegeben werden.

Netzwerktest

Werden heute meist manuell oder mit Hilfe eines Skripts durchgeführt. Ein automatisierter Netzwerktest sollte hypothetisch ad-hoc nach jeder Konfigurationsänderung vom Testprogramm durchgeführt werden um die Funktionsweise des Netzwerks zu validieren. Ein Benutzer des zu entwickelnden Systems soll in der Lage sein, mit nur geringer Einarbeitungszeit, Netzwerktests zu spezifizieren und durchzuführen.

2.3 Use Cases

2.3.1 Use Case Diagramm

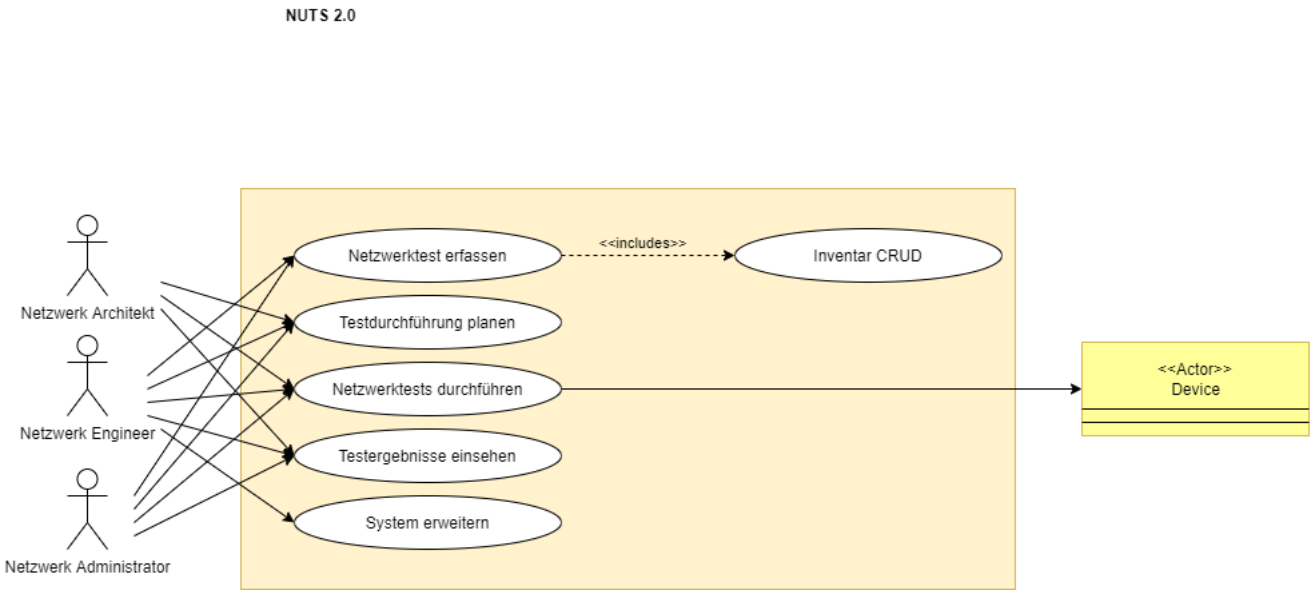

Abbildung 1: Use Case Diagramm

2.3.2 Aktoren

Die Primären Akteure sind der Netzwerk Architekt, -Administrator und -Engineer. Der Architekt will primär die Ergebnisse einsehen können, um zu sehen, dass das Netzwerk korrekt funktioniert. Ausserdem möchte er, beispielsweise um eine Erweiterung des Netzwerks zu Planen, eine Ausführung von Netzwerktests konfigurieren und durchführen. Der Administrator will Netzwerktests erfassen, deren Durchführung planen, die Tests durchführen und die Ergebnisse einsehen. Der Engineer möchte neben den Tätigkeiten, die der Administrator ausführt, zudem das Testsystem um weitere Netzwerktests erweitern können.

2.4 Beschreibung Usecases (Brief)

Netzwerktest erfassen Ein Netzwerktest setzt sich zusammen aus der Testdefinition mit den zu testenden Devices, Befehle, die auf den Devicesv ausgeführt werden sollen, einem oder mehreren Kommunikationskanälen, über den die Devices angesprochen werden und einem Erwartungswert für das Ergebnis. Diese Informationen werden in einer Testdefinitionssprache gespeichert, welche so strukturiert sein muss, dass sie ein Softwaresystem einfach laden kann und trotzdem von Menschen interpretiert werden kann. Die Erfassung eines Netzwerktests soll so einfach wie möglich gehalten werden, damit Administratoren und Engineers effizient neue Tests spezifizieren können.

Inventar CRUD Der Netzwerk Engineer oder -Administrator möchte die physischen und virtuellen Netzwerkgeräte und deren statische Konfiguration in einem Inventar verwalten. Das Inventar wird von dem zu entwickelnden System verwendet, um die Gerätekonfigurationen wie z.B. Zugangsdaten oder Herstellerinformationen abzurufen. Es soll möglich sein, das Inventar automatisiert zu erstellen und Geräte in distinkte Gruppen zu kategorisieren.

Testdurchführung planen Themen, die in der Testausführung relevant sind, sind die Auswahl, welche Tests überhaupt ausgeführt werden sollen, die Reihenfolge der Tests, auf welchen Teil des Systems sie angewandt werden sollen und ob sie synchron oder asynchron durchgeführt werden. Weitere Punkte wären das automatische durchführen von Tests zu spezifischen Zeiten oder Wochentagen und dass die Testkonfiguration gespeichert wird, um sie später anzupassen. Auch hier ist darauf zu achten, dass das zu entwickelnde System so aufgebaut ist, dass Engineers, Architekten und Administratoren effizient arbeiten können.

Netzwerktests durchführen Der Anwender möchte die in der Testdurchführung geplanten Netwerktests auf das angegebene System ausführen. Dazu wird in einer Benutzeroberfläche die Testausführung gestartet oder auf einem Gerät automatisch die zu entwickelnde Software ausgeführt.

Testergebnisse einsehen Nachdem ein Test durchgeführt wurde, soll ein Testresultat angezeigt werden. In den Resultaten soll ersichtlich sein, welche Tests durchgeführt wurde, was der Test genau gemacht hat, welche Befehle auf welchen Devices ausgeführt wurde und wie das Ergebnis ist. Die Testergebnisse sollen auf der Konsole/Benutzeroberfläche ersichtlich sein und zusätzlich in einem Testreport mit Datum und Uhrzeit gespeichert werden, damit die Historie des Netzwerks ermittelt werden kann. Wenn Tests durch einen Fehler im zu entwickelnden System nicht durchgeführt werden kann, sollen alle anderen Tests nicht davon beeinflusst werden und das Ergebnis soll einen Vermerk für das Versagen des Systems beinhalten, mit dem der Anwender in der lage ist, die Ursache zu ermitteln und zu beheben.

System erweitern Engineers sollen in der Lage sein, das System bei Bedarf zu erweitern, z.B. um weitere Tests oder Netzwerkschnittstellen hinzuzufügen oder Fehler zu verbessern. Die Erweiterungen beschränken sich aber auf rein funktionale Bereiche des Systems. Für Änderungen an der Benutzeroberfläche sollen Softwareengineers mit Erfahrung auf dem Gebiet hinzugezogen werden.

2.5 Nichtfunktionale Anforderungen

In diesem Kapitel werden die nichtfunktionalen Anforderungen an das Projekt behandelt. Es werden Aspekte und Anforderungen aus den Bereichen Änderbarkeit, Benutzbarkeit, Effizienz, Zuverlässigkeit, Betreibbarkeit und Sicherheit gemäss ISO/IEC 9126 betrachtet. Die jeweiligen Aspekte werden in ihren Unterkapiteln genauer beschrieben. Es wurden mögliche Szenarien erarbeitet, die in der Erstellung oder dem Betrieb der Software auftreten können und beim Architekturdesign in betracht gezogen wurden.

2.5.1 Änderbarkeit

Aufwand, der zur Durchführung von vorgegebenen Änderungsarbeiten benötigt wird. Unter Änderungen gehen Korrekturen, Anpassungen oder Veränderungen der Umgebung, Anforderungen oder funktionalen Spezifikation. Gemäss ISO 9126 gehören zur Änderbarkeit folgende Teilmerkmale:

Analysierbarkeit Aufwand, der benötigt wird, um das System zu verstehen, z.B. um Ursachen von Versagen oder Mängel zu diagnostizieren oder Änderungen zu planen.

Modifizierbarkeit Wie leicht lässt sich das System anpassen, um Verbesserungen oder Fehlerbeseitigungen durchzuführen.

Stabilität Wahrscheinlichkeit, dass mit Änderungen unerwartete Nebenwirkungen auftreten.

Testbarkeit Wie gross wird der Aufwand, bei Änderungen die Software zu prüfen.

Szenario: Neue Netzwerkschnittstelle Wenn zum bestehenden System eine neue Netzwerkschnittstelle definiert werden soll, so muss die dafür notwendige Software innerhalb von einer Arbeitswoche entwickelt, integriert und in Betrieg genommen werden können.

Qualitätsziele	Flexibilität, Erweiterbarkeit, Anpassbarkeit, Austauschbarkeit
Geschäftsziel(e)	Software kann mit geringem Aufwand an geänderte Anforderungen angepasst werden
Auslöser	Ein Engineer möchte weitere Tests einbinden oder Schnittstellen, die nicht im System integriert sind.
Reaktion	Die Software lässt sich von einem Entwickler in weniger als einer Woche um benötigte Komponenten erweitern.
Zielwert	Erweiterungen der Netzwerkschnittstellen oder Anpassungen von Tests sind innerhalb von 40 Personenstunden umsetzbar.

Tabelle 1: Szenario: Neue Schnittstelle

Scenario: Schnelle Fehlerlokalisierung Die Ursache von fehlgeschlagenen Tests (Software-Unittests) lässt sich in kurzer Zeit lokalisieren.

Qualitätsziele	Schnelle Fehlerbehebung, Änderbarkeit, Anpassbarkeit, geringes Risiko bei Erweiterungen
Geschäftsziel(e)	Entwickler können das Programm einfach anpassen und erkennen im Fehlerfall schnell, was nicht funktioniert hat.
Auslöser	Eine Änderung im Code führt zu Fehlnern in der Ausführung.
Reaktion	Wenn ein Fehler dazu führt, dass die Softwareausführung fehlschlägt, kann ein Entwickler aufgrund von Fehler- und/oder Log-Nachrichten die Ursache in kurzer Zeit lokalisieren.
Zielwert	Fehlerlokalisierung findet durchschnittlich in weniger als 10 Minuten statt.

Tabelle 2: Szenario: Schnelle Fehlerlokalisierung

2.5.2 Benutzbarkeit

Zeitlicher Aufwand, der für die Erlernung der Benutzung des Programms benötigt wird. Die User werden hierfür in spezifische Nutzergruppen mit festgelegten Fähigkeiten unterteilt.

Verständlichkeit Aufwand für den Nutzer, die Konzepte und Menüführung der Anwendung zu verstehen.

Erlernbarkeit Aufwand für den User, sich ohne Vorwissen in das System einzuarbeiten.

Bedienbarkeit Aufwand für den Benutzer, die Anwendung zu bedienen.

Szenario: Einfachheit der Definitionssprache Die Definitionen von Inventar und Tests sind so aufgebaut, dass ein User in kurzer Zeit die Struktur und den Aufbau versteht und eigene Tests implementieren kann.

Qualitätsziele	Produktivität, Einfachheit, Verständlichkeit
Geschäftsziel(e)	Einarbeitung in die Testdefinition erfolg möglichst einfach und benötigt nur geringes Vorwissen.
Auslöser	Ein Nutzer, welcher keine Erfahrung im Umgang mit der Software hat, möchte eigene Tests definieren.
Reaktion	Benutzer können sich schnell in die Testdefinitionen einlesen und rasch eigene Tests definieren, vorausgesetzt, sie haben Kenntnisse des Netzwerkes.
Zielwert	Ungeschulte Nutzer verstehen innerhalb von durchschnittlich 30 Minuten die Struktur und den Aufbau der Testdefinitionen und sind in der Lage, eigene Tests zu erstellen.

Tabelle 3: Szenario: Einfachheit der Definitionssprache

Szenario: Hinweis auf Fehleingaben Fehlerhafte Eingaben werden vom System ignoriert und der Benutzer wird auf die falsche Eingabe hingewiesen. Das Programm führt fehlerfreie Programmteile unabhängig von den Fehlern durch.

Qualitätsziele	Robustheit, Verständlichkeit, Fehlertoleranz.
Geschäftsziel(e)	Fehleingaben führen nicht dazu, dass die Tests nicht mehr durchgeführt werden können.
Auslöser	Ein Benutzer macht einen Fehler bei der Testdefinition und startet das Programm.
Reaktion	Das Programm führt alle korrekten Tests durch und informiert den Benutzer, dass es fehlerhafte Tests gibt, die nicht durchgeführt werden können. Die Hinweise werden im Report und auf der Konsolenausgabe geschrieben.
Zielwert	Tests sind einzeln gekapselt und werden unabhängig voneinander durchgeführt. Falscheingaben werden vom Programm detektiert und im Testreport sowie auf der Konsolenausgabe erwähnt.

Tabelle 4: Szenario: Hinweis auf Fehleingaben

2.5.3 Effizienz

Mit Effizienz ist die 'performance efficiency' gemeint, d.h. das Verhältnis zwischen dem Leistungsniveau der Software und den eingesetzten Hardwarekomponenten. Andere Beschreibungen umfassen: Skalierbarkeit, Speicherbedarf, Verarbeitungsgeschwindigkeit, Antwortzeit etc. Teilmerkmale nach ISO 9126:

Zeitverhalten Dauer für Verarbeitung und Antwortzeit sowie Durchsatz bei der Ausführung des Programms

Verbrauchsverhalten Wie viel Speicherbedarf hat das Programm, wie lange werden Betriebsmittel in Anspruch genommen und welche Hardwarekomponenten werden benötigt.

2.5.4 Zuverlässigkeit

Unter Zuverlässigkeit versteht man die Fähigkeit der Software, unter festgelegten Bedingungen die Funktionalität über einen definierten Zeitraum zu gewährleisten

Reife Geringe Ausfallhäufigkeit durch Fehlzustände.

Fehlertoleranz Die Software ist in der Lage, trotz Fehlern ihr spezifiziertes Leistungsniveau beizubehalten.

Wiederherstellbarkeit Im Fehlerfall können betroffene Daten wiederhergestellt und die Funktionalität wieder aufgenommen werden.

Szenario: Tests lassen sich auf der Netzwerkseite nicht ausführen Falls ein Test auf dem jeweiligen Netzwerkgerät nicht erfolgreich durchgeführt werden kann, läuft das Programm weiter und definiert den dazugehörigen Netzwerktest als nicht bestanden.

Qualitätsziele	Robustheit, Behandlung Infrastrukturbedingter Fehler.
Geschäftsziel(e)	Das System führt alle Tests unabhängig voneinander durch. Wenn ein Test zu einem Fehler führt, weil z.B. ein falsches Netzwerkgerät angegeben wurde, wird dieser Test unabhängig von allen anderen Tests fehlschlagen.
Auslöser	Test lässt sich auf spezifizierter Infrastruktur nicht ausführen.
Reaktion	Test schlägt fehl und mögliche Ursachen werden im Report und in der Konsole angezeigt. Alle anderen Tests laufen durch.
Zielwert	Das Fehlschlagen eines Tests fürht nicht zum Programmabbruch.

Tabelle 5: Szenario: Testausführung Netzwerkseitig nicht ausführbar

2.5.5 Betreibbarkeit

Die Betriebbarkeit wird in der ISO 9126 nicht definiert. Die ISO spezifiziert aber mehrere Teilmerkmale, die unter dem Begriff Betreibbarkeit zusammengefasst werden können:

Analysierbarkeit Aufwand, der benötigt wird, um den Code zu analysieren, um im falle eines Versagens dessen Ursachen zu diagnostizieren oder um Änderungen zu planen und durchzuführen.

Installierbarkeit Aufwand, das Programm auf einem frisch aufgesetzten Gerät laufen zu lassen.

Übertragbarkeit Kann die Software von einer Umgebung auf eine andere übertragen werden. Als Umgebung zählen Hardwarekomponenten, Softwarekomponenten, Organisatorische Umgebungen oder Betriebssysteme.

Austauschbarkeit Aufwand und Möglichkeit, die Software anstelle einer anderen in deren spezifizierten Umgebung laufen zu lassen.

Koexistenz Fähigkeit der Software, neben anderen Programmen mit ähnlichen oder übereinstimmenden Funktionen zu arbeiten.

Szenario: Einfache Installation auf einem neuen Gerät Das Programm lässt sich auf einem neuen Gerät ohne grossen Mehraufwand installieren, ohne dass die Funktionalität des Geräts beeinflusst wird.

Qualitätsziele	Einfachheit, Portierbarkeit, Benutzbarkeit
Geschäftsziel(e)	Die Installation der Software ist so einfach, dass sie innert kurzer Zeit und/oder automatisiert durchgeführt werden kann.
Auslöser	Die Testsoftware soll auf einem frisch aufgesetzten Gerät installiert werden.
Reaktion	Installationszeiten sind gering, benötigen wenige bis keine weiteren Softwarekomponenten oder lässt sich mit einigen Kommandozeilenbefehlen automatisch installieren.
Zielwert	Die Software wird mit einer Installationsanleitung ausgeliefert, die einfach und verständlich die Inbetriebnahme des Programms erklärt. Abhängigkeiten zu anderen Softwarekomponenten werden bewusst gering gehalten um eine einfache Installation mit weniger als 30 Minuten Zeitaufwand zu gewährleisten.

Tabelle 6: Szenario: Einfache Installation auf einem anderen Gerät

2.5.6 Sicherheit

In dieser Sektion werden Sicherheitsanforderungen beschrieben. Verschlüsselung, Privacy und der Umgang mit Passwörtern.

Verschlüsselung von Datenübertragungen Die Netzwerktest werden über eine Verschlüsselte Verbindung durchgeführt, die dem aktuellen Stand der Technik entspricht.

Umgang mit Passwörtern Zugangsdaten der Devices werden im Inventar in Unverschlüsselter Form abgelegt. Es liegt in der Verantortung der Betreiber des Netzwerks, dass die Zugangsdaten nicht von dritten eingesehen werden.

3 Design

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

4 Realisierung

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

5 Zeitauswertung

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

6 Be(nuts)eranleitung

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

7 Persönliche Berichte

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

8 Eigenständigkeitserklärung

Die Authoren erklären hiermit,

- dass die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt wurde, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit den Betreuerm vereinbart wurde.
- dass sämtliche verwendeten Quellen erwähnt und gemäss den gängigen wissenschaftlichen Zitierregeln korrekt angegeben wurden.
- dass keine durch Uhrheberrecht geschützten Materialien (z.B. Bilder) in dieser Arbeit in unerlaubter weise verwendet wurden.

Ort, Datum:	Ort, Datum:
Name, Unterschrift:	Name, Unterschrift:

9 Vereinbarung Urheber-/ Nutzungsrechte

1. Gegenstand der Vereinbarung

Mit dieser Vereinbarung werden die Rechte über die Verwendung und die Weiterentwicklung der Ergebnisse der Studienarbeit NUTS2.0 von Janik Schlatter und Mike Schmid unter der betreuung von Beat Stettler und Urs Baumann geregelt

2. Urheberrecht

Die Urheberrechte stehen den Authoren zu.

3. Verwendung

Die Ergebnisse der Arbeit dürfen sowohl von den Authoren, von der OST (ehemals HSR), sowie vom INS Institut for Networked Solutions nach Abschluss der Arbeit verwendet und weiter entwickelt werden.

4. Softwarelizenz

Die in der Arbeit entstandene Software untersteht der Apache Licence 2.0. Somit darf die Software frei verwendet, verändert und verteilt werden, insofern dies nicht die in der Lizenz genannten Regelungen verletzt.

Rapperswil, den..... .....
Die Studentin/ der Student

Rapperswil, den..... .....
Die Studentin/ der Student

Rapperswil, den..... .....
Der Betreuer/ die Betreuerin der Studienarbeit

Rapperswil, den..... .....
Der Betreuer/ die Betreuerin der Studienarbeit