Anforderungsanalyse

Studienarbeit FS-2020

19. März 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

1 Sinn und Zweck

Dieses Dokument beschreibt die Anforderungen an die Studienarbeit Nuts 2.0. Es umfasst eine Übersicht über die Problemdomäne, die Requirements-Analyse und die nichtfunktionalen Anforderungen an das zu entwickelnde System.

Änderungsgeschichte

Datum	Version	Änderung	Autor
03.03.2020	1.0	Initial Setup	Janik Schlatter
11.03.2020	1.0	Fertigstellung für Meilenstein: Requirements	Janik Schlatter, Mike Schmid
12.03.2020	2.0	Revision gem. Beschprechung vom 11.03.2020	Janik Schlatter, Mike Schmid

Inhaltsverzeichnis

2 Übersicht Problemstellung

2.1 Generell

Bei der Entwicklung von Netzwerkumgebungen werden auch in modernen Systemen die Überprüfungen und Tests der Konfigurationen meistens von Hand vorgenommen. Die Applikation soll ein Framework zur Verfügung stellen, mit dem Netzwerke automatisiert getestet werden können, vergleichbar mit Unit-Tests in der Software-Entwicklung. Die folgenden Abschnitte sollen aufzeigen, welche Herausforderungen an ein solches System gestellt werden und wie ein Framework damit umgehen könnte. Dazu wurden die Kern-Akteure in einem Netzwerk identifiziert und deren Funktionalität und gegenseitige Abhängigkeiten so generalisiert wie möglich formuliert, um auf dieser Basis die Testsoftware zu entwerfen. Es wurden Annahmen bezüglich der Namensgebung getroffen. Beispielsweise sind die Bezeichnungen der einzelnen Netzwerk-Techniker überall ein wenig anders formuliert.

2.2 Akteure im aktuellen System

Die hier beschriebenen Akteure sind in einem herkömmlichen Netzwerksystem anzutreffen.

Akteur	Beschreibung
Netzwerk-Architekt	Ein Netzwerk-Architekt plant und erstellt Kommunikationsnetzwerke. In der Praxis oft auch als Network-Engineer bezeichnet. 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	Der 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 Netz-Admin wenige bis keine 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 kennen.
Netzwerk-User	Benutzer der Netzwerkumgebung. User müssen das Netzwerk verwenden, aber nicht dessen Konfigurationen anpassen können.
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 Zustand zur Laufzeit. In den kommenden Kapiteln wird genauer auf Netzwerkgeräte eingegangen.
Repository/Inventar	Im Inventar werden die Unterschiedlichen Devices mit den für den Betrieb wichtigsten Parametern abgelegt. Das Inventar kann in digitaler Form als Repository, als File auf einem Ordner/Computer, oder analog in einem Order abgelegt sein. Das Inventar wird benötigt, um die aktuellen Konfigurationen, die physische Position des Geräts oder sonstige für den Betrieb relevanten Informationen zu dokumentieren.

Tabelle 1: Akteure in einem Netzwerksystem

2.3 Zusätzliche Entitäten im zu entwickelnden System

Diese Entitäten müssen zusätzlich zu den im Abschnitt 2.2 beschriebenen Akteuren in einem Netzwerk, welches ein System für das automatisierte Testen beinhaltet, auftreten.

Testprogramm	Das Testprogramm ist das zu entwickelnde System in dieser Arbeit. Es interagiert mit den anderen Akteuren und hat, abhängig von Akteur und Kontext, unterschiedliche Anforderungen.
Testdefinitionssprache	Kann auch Testbeschreibungssprache genannt werden. Testbeschreibungen sollten zwischen dem Systemmodell und den low-level-Testcases angelegt werden und möglichst einfach und allgemein aufgebaut sein. Eine Testdefinition beschreibt unabhängig von der zu verwendeten Programmiersprache oder ausführenden Plattform die einzelnen Testfälle.
Testreport	Ausgabewerte der durchgeführten Tests. Reports sollten so strukturiert sein, dass ein Netzwerk-Techniker mit wenig Aufwand erkennen kann, welche Tests erfolgreich verlaufen sind, welche nicht erfolgreich waren und was mögliche Ursachen dafür waren. Testreports können in der Dokumentenablage des ausführenden Systems oder in einem Repository abgelegt werden und benötigen neben den Testfällen im Minimum noch ein Durchführungsdatum und -Uhrzeit.
Kommunikationskanal	Der Kommunikationskanal verbindet das zu testende Netzwerk mit dem Testprogramm. Abhängig vom Netzwerk geschieht dies über Kabelverbindungen oder Kabellos. Es gibt verschiedene Technologien, die die Kommunikation über das Medium ermöglichen und je nach Schnittstelle unterschiedliche Ergebnisse und -formate liefert.
Netzwerktest	Werden meistens von Hand oder über Scripts ausgeführt. Ein automatisierter Netzwerktest sollte hypothetisch ad-hoc nach jeder Konfigurationsänderung durchgeführt werden um zu validieren, dass das Netzwerk noch wie gewollt läuft.

Tabelle 2: Entitäten im zu entwickelnden System

2.4 Herausforderungen

Das Testen von Netzwerken ist ein komplexes Unterfangen. Der Netzwerkzustand ändert sich zur Laufzeit dauernd, um sich an Änderungen von einzelnen Geräten anzupassen. User treten dem Netzwerk bei oder verlassen dieses und Netzwerkgeräte passen über dynamische Protokolle die Verbindungen an um eine möglichst performante Verbindung zu ermöglichen oder um Fehler/Ausfälle zu korrigieren. Weiterhin können spezielle Konfigurationen wie die Priorisierung von verschiedenen Kommunikationsarten z.B. Internettelefonie (VoIP), auch Quality of Service (QoS) genannt, im System vorkommen, was die genaue Erfassung des Ist-Zustandes noch komplizierter macht. In diesem Kontext muss ein Netzwerk-Testsystem nicht nur die Gerätekonfiguration beim Starten, sondern auch die dynamische Laufzeitkonfiguration berücksichtigen.

2.5 Beschreibung Software-Unit-Testing

Das Testen von Software kann grob in zwei Kategorien unterteilt werden, statisch und dynamisch. Darüber hinaus gibt es weitere Unterteilungen, je nachdem, was in welchem Umfang wie getestet werden soll. Die Unittests sind dabei in der dynamischen Kategorie angesiedelt und dienen der Verifikation der Software-Implementation.

Statische Tests umfassen Code- und Design-Review, Code-Guidelines und formale Methoden.

Dynamische Tests sind üblicherweise Softwarebestandteile, die andere Komponenten mit verschiedenen Methoden auf die korrekte Ausführung testen.

2.5.1 Anforderungen an Softwaretests

Softwaretests müssen folgende Anforderungen erfüllen:

Die Tests müssen geplant sein und es muss ein Test Plan existieren.

Tests müssen systematisch spezifiziert sein.

Die Resultate der Tests müssen dokumentiert werden.

Nach Möglichkeit sollen Tests automatisch ausgeführt werden.

Tests müssen reproduzierbar und nachvollziehbar sein, d.H. sie müssen sich im Debugger Schritt für Schritt durchführen lassen.

3 Detailbeschreibung Akteure

Dieser Abschnitt beschreibt die Akteure, welche im Abschnitt 2.2 genannt werden und ihre spezifischen Anforderungen an ein Testsystem sowie Einschränkungen, die vom System auf die Akteure gelten.

3.1 Personen

Akteure interagieren mit dem zu entwickelnden System und hat in dessen Kontext eine Rolle und Ziele. Akteure lassen sich in drei Kategorien unterteilen:

Primärer Aktor	Person oder Objekt, dessen ziele durch Interaktion mit dem zu entwickelnden System erfüllt werden.
Unterstützender Akteur	Bietet dem System Services an, z.B. Techniksupport eines Herstellers.
Nebenaktor	Hat Interesse am Verhalten des Systems oder zieht einen Nutzen daraus, ist aber nicht primär oder unterstützend.

Tabelle 3: Kategorien der Akteure

3.1.1 Netzwerk-Architekt

Rolle

Primärer Aktor

Ziele

Konfiguration automatisierter Tests (welche er für analytische Zwecke benötigt) Durchführung der Tests, Sichten der Resultate

3.1.2 Netzwerk-Engineer

Rolle

Primärer Aktor

Ziele

Verwalten von Devices in einem Inventar, Konfiguration automatisierter Tests, Durchführung der Tests, Sichten der Resultate Erweiterung des Testsystems

3.1.3 Netzwerk-Administrator

Rolle
Primärer Aktor

Ziele
Verwalten von Devices in einem Inventar, Konfiguration automatisierter Tests (einfachere Tests, die für einen Administrator wichtig sind), Durchführung der Tests, Sichten der Resultate

3.1.4 User

Rolle
Nebenaktor

Ziele
Verwenden des Systems

3.1.5 Einschränkungen

Prioritärer Aktor
Das zu entwickelnde System wird primär mit dem Netzwerk-Engineer als Aktor aufgebaut, da der Architekt eher eine planende/managende Rolle einnimmt und der Administrator für einfachere Aufträge, wie das umschalten von Ports oder Interfaces, zusständig ist.

User
Für das zu entwickelnde System hat der User keinen Einfluss auf die Funktionalen Anforderungen. Er ist darauf angewiesen, dass das Netzwerk möglichst schnell und fehlerfrei funktioniert. In den Nichtfunktionalen Anforderungen wird auf diese Bedürfnisse eingegangen.

Kenntnisse der Netzwerk-Techniker
Um Tests auf dem Netzwerk effektiv durchführen zu können, benötigen die primären Akteure einige Fähigkeiten:

Sie müssen wissen, wie sich das Netzwerk zur Laufzeit verhalten soll.

Tester müssen Benutzeraktivitäten gegenüber dem Netzwerk imitieren können.

Falls gewisse Devices nicht getestet werden können, müssen Tester deren Funktionalitäten bei Bedarf gegenüber dem Netzwerk imitieren können.

Sie müssen wissen, was das Netzwerk als Reaktion auf ihre Aktivitäten macht.

Tester müssen in der Lage sein, den Fortschritt und die Resultate der Tests zu interpretieren und die Ursachen von gescheiterten Tests ausfindig machen.

3.2 Devices

Um Tests durchführen zu können, werden mehrere Devices benötigt, die in einer bestehenden Konfiguration in einem Netzwerk integriert sind. Die Devices müssen dafür einen oder mehrere Kommunikationskanäle bereitstellen, über die man die Konfiguration abfragen und Befehle senden kann. Um einen Netzwerktest Schritt-für-Schritt durchführen zu können, werden verschiedene Zwischenschritte benötigt, welche die Anforderungen and das zu entwickelnde System darstellen. Geräte müssen mit ihrer Start-Konfiguration in einem Inventar erfasst werden, die Startkonfiguration und Laufzeitkonfiguration muss abrufbar und speicherbar sein, und die Geräte sollten sich in logische Gruppen, wie beispielsweise Aufteilung nach Gebäude oder nach VLAN (Virtual Local Area Network), kategorisieren lassen.

3.2.1 Devices erfassen

Das Erfassen von Geräten beinhaltet das Abrufen der Gerätekonfiguration und das Speichern dieser Konfiguration in einem Inventar. Das Inventar kann mit einer Beschreibungssprache wie XML, YAML, YANG oder weitere, geführt werden, die es erlaubt, die Informationen dynamisch in das zu entwickelnden System zu laden und für die Testdefinition zu verwenden. Die Erfassung der Devices kann dabei in folgende Stufen unterteilt werden, die mit steigender Automatisierung ein komplexeres System voraussetzen.

Stufe	Beschreibung
Stufe Manuell	Die Devices werden vom Benutzer manuell in einem File hinzugefügt welches optimalerweise in einem Repository gespeichert wird.
Stufe Formular	Der Netzwerk-Techniker kann mit Hilfe eines Formulars oder einem Grafikinterface des zu entwickelnden Systems die Devices erfassen. Da dies über ein Formular geschieht, muss der Benutzer nicht manuell für jedes Device ein neues File anlegen, sondern das Programm erledigt all das für den Benutzer und er kann sich das Inventar im GUI zusammenklicken.
Stufe Automatisiert	Die Devices werden automatisch von dem zu entwickelnden System aus dem Netzwerk ausgelesen und in ein Inventar hinzugefügt. Das Ergebnis kann vom Techniker dann gesichtet werden und lässt sich bei Bedarf anpassen.

Tabelle 4: Stufen der Device-Erfassung

3.2.2 Device Informationen

Je spezifischer die Tests des zu entwickelnden Programms sind, desto mehr Informationen eines einzelnen Devices werden benötigt. Die Informationen der Devices werden in Kategorien unterteilt: Minimaleinstellungen , Erweiterte Einstellungen und Alle Informationen. Je mehr Informationen von einem Gerät im Inventar abgelegt sind, desto mehr Tests lassen sich ohne Erfassung von zusätzlichen Infos durchführen.
Die folgende Tabelle zählt einige Beispiele dieser Einstellungen auf und beschreibt, wozu die Informationen benötigt werden.

Einstellung	Beschreibung
Name und Passwort	Diese Informationen werden benötigt, um sich bei einem Device (z.B. Router) anzumelden, um dort Befehle für die Tests ausführen zu können.
MAC-Adresse	Die Media Access Control Adresse ist für die eindeutige Identifikation eines Netzwerkgerätes. Die MAC kann z.B. für die Berechnung von IPv6 Adressen verwendet werden.
IP Adresse	Die IP-Adresse eines Devices ist die Virtuelle Adresse, die das Gerät im Netzwerk hat.
Seriennummer	Die Seriennummer eines Geräts wird nur für sehr spezielle Anwendungen benötigt z.B. die Entscheidung des Wurzelknotens im Zweifelsfall beim Spanning Tree Protokoll.
Statische Routen	Statische Routen sind manuell konfigurierte Einstellungen, die nicht verändert werden, wenn das Netz nicht angepasst wird. Sie sind einfach zu konfigurieren, aber sie können sich nicht an Systemfehler anpassen.
Dynamisches Routing	Routing Protokolle passen die konfigurierten Routen der Geräte zur Laufzeit an, um auf Änderungen im Netz zu reagieren. Dynamische Protokolle sind flexibler, aber auch komplexer als statische Routen.
Virtual LAN	Das Virtuelle Local Area Network unterteilt Geräte im Netzwerk in logische Gruppen und bewirkt eine Segmentierung des Netzwerks was zu erhöhter Sicherheit, Erweiterbarkeit und besserem Netzwerkmanagement führen kann.
Overlay Netzwerk	In einem Overlay Netzwerk werden logische Verbindungen von der virtuellen Konfiguration getrennt. In jedem Layer nimmt ein physisches Gerät dabei eine andere Rolle wahr, was dazu führt, dass die Gerätekonfiguration schwieriger zu testen sind.

Tabelle 5: Beispiele für Device-Informationen

3.2.3 Device Gruppen

Die Devices sollen sich in logische Gruppen unterteilen lassen. Somit könnte man zum Beispiel alle Devices in einem Gebäude in die Gruppe: 'Gebäude 1' schieben. Im Kontext des zu entwickelnden Systems würde man damit die Möglichkeit haben, Filter auf die Geräte anzuwenden, um nur spezifische Einstellungen zu testen.

3.3 Netzwerktest

Netzwerktest werden von den Technikern in einer Testbeschreibungssprache erfasst und vom zu entwickelnden System automatisiert ausgeführt. Dabei werden verschiedene Anforderungen an eine Testbeschreibung gestellt. Zum Beispiel sollten sich Tests in logische Kategorien unterteilen lassen, die Erfassung der Tests soll möglichst automatisch durchführbar sein und die Ergebnisse der Tests müssen verständlich sein und sich in einem Report speichern lassen.

3.3.1 Test Kategorien/Typen

Die Tests werden grundsätzlich nach Kategorien unterschieden, beispielsweise sind alle Ping-Tests eine Art. Die Einteilung von Tests in Kategorien und Typen erlaubt es dem Tester, diese Tests im System zu filtern und Tests kategorisch auszuwählen, wenn er eine Testausführung plant. Ein wichtiger Anforderungspunkt ist dabei, dass Techniker in Zukunft möglichst einfach weitere Kategorien/Testarten hinzufügen können.

3.3.2 Test erfassen

Auch hier gibt es wieder mehrere Stufen, wie die Tests erfasst werden können.

Stufe	Beschreibung
Stufe Manuell	Der Benutzer erfasst die Tests komplett manuell in einem File mit einer Testdefinitionssprache. In diesem File muss der Benutzer die Testart, die beteiligten Devices und weitere Optionen des Testes festlegen. Ausserdem ist das zu erwartende Ergebnis für einen Test zu spezifizieren.
Stufe Formular	Das Programm stellt ein Formular zur Verfügung, welches die ganze Testerfassung vereinfacht. Der Benutzer kann sich hierbei den Test nur noch über die Benutzeroberfläche zusammenklicken. Das Programm speichert danach den Test automatisch in einem File oder Repository.

Tabelle 6: Stufen der Test-Erfassung

3.3.3 Erwartungswert erfassen

Ein Test findet in der Regel durch einen Vergleich vom Ist-Wert mit dem Soll-Wert statt. Der Ist-Wert in einem Netzwerk ist die Konfiguration und der Zustand zur Laufzeit. Der Sollwert muss vom Netzwerk-Techniker bei der Testdefinition erfasst werden.

3.3.4 Testreihenfolge

Es kann sein, dass vom Tester eine gewisse Reihenfolge für die Ausführung der Tests gewünscht ist. Dafür sollten sich Tests nach Kategorie (Ping, Traceroute etc.), nach Layer, nach Gerätetyp (Switch, Router, Server, etc.) oder nach eigenen Kategorien (z.B. nur Gebäude 1), filtern lassen. Die Auswahl der Testreihenfolge soll dabei möglichst intuitiv mit einer grafischen Benutzeroberfläche geschehen und mit wenigen Klicks durchführbar sein.

3.3.5 Test Durchführung

Wenn der Benutzer die Durchführung startet kann er mit Hilfe der Test-Gruppen die Reihenfolge der Tests bestimmen. Der Benutzer kann zudem noch bestimmen welche Tests synchron und welche asynchron durchgeführt werden. Das Programm lädt aus der Testdefinition und dem Inventar die benötigten Informationen und greift danach über die Netzwerkschnittstelle auf die Devices zu und führt die Tests in der angegebenen Reihenfolge durch. Die vom Netzwerk zurückgegebenen Werte werden mit den Erwartungswerten verglichen und es wird darüber entschieden, ob ein Test bestanden oder nicht bestanden ist. Die Ergebnisse werden danach direkt über das Terminal oder GUI angezeigt und zusätzlich in einem Testreport gespeichert.

3.4 Netzwerkschnitstelle

Jedes Netzwerkgerät hat eigene Methoden, wie man darauf zugreift und welche Werte für unterschiedliche Befehle zurückgegebenen werden. Die Auswahl der Netzwerkschnittstelle soll dabei optimalerweise dem Netzwerk-Techniker überlassen werden. Das zu entwickelnde System zeigt dafür die zur Verfügung stehenden Schnittstellen an und der Techniker wählt daraus eine aus. Es wird eine empfohlene Default-Schnittstelle benötigt, um Technikern mit geringer Erfahrung die Möglichkeit zu bieten, die Auswahl zu umgehen.

3.5 Logging

Die Auswertung der jeweiligen Durchführung der Tests wird in einem Testreport gespeichert, welcher der User jederzeit einsehen kann, um sich einen Überblich über die Historie vergangener Testdurchführungen zu verschaffen. Diese Testreports werden in einem Repository gespeichert und benötigen ein Durchführungsdatum und -Zeit.

4 Use Cases

4.1 Personas

Personas lassen sich in zwei Gruppen aufteilen. Der Netzwerk-Architekt und Netzwerk-Engineer sind eine Gruppe. Sie sind in der Lage, Python Code zu verstehen und anzupassen. Sie werden in den Personas als Engineer zusammengefasst. Der Netzwerk-Administrator gehört zur zweiten Gruppe und wird in den Personas als Administrator bezeichnet. Es wird nicht von ihm erwartet, dass er am Python-Code des zu entwickelnden Systems etwas verändert. Der Benutzer des Netzwerks bekommt keine eigene Perona. Er ist daran interessiert, das Netzwerk zu verwenden, nimmt aber auf dessen Funktionen keinen direkten Einfluss.

Person	Beschreibung	Technisches Wissen
Engineer	Hat fundierte Kenntnisse über das Netzwerk und dessen Zustände	Er versteht Python-Code und kann das zu entwickelnde System seinen Bedürfnissen anpassen.
Administrator	Setzt Änderungen am Netzwerk um und hat Kenntnisse über die ihm zugeteilten Bereiche.	Es wird nicht davon ausgegangen, dass ein Administrator genügend Python-Kenntnisse hat, um am System etwas zu verändern.

Tabelle 7: Personas

4.2 Use Cases Brief

Die Use-Cases beschreiben die Funktionalen Anforderungen an das zu entwickelnde System und dessen Komponenten.

4.2.1 Inventar CRUD

Der Engineer oder der Administrator möchte die physischen und virtuellen Geräte und deren Konfigurationen in einem Inventar verwalten. Das Inventar wird vom zu entwickelnden System verwendet um die Gerätekonfigurationen wie Zugangsdaten oder Herstellerinformationen abzurufen. Es soll möglich sein, das Inventar möglichst automatisiert zu erstellen und Geräte in distinkte Gruppen zu kategorisieren.

4.2.2 Netzwerktest erfassen

Ein Netzwerktest setzt sich zusammen aus den zu testenden Devices, Befehle, die auf den Devices ausgeführt werden, einem oder mehreren Kommunikationskanälen 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 soll trotzdem von Menschen interpretiert werden können. Die Erfassung eines Netzwerktests soll so einfach wie möglich gehalten werden, damit Administratoren und Engineers effizient neue Tests spezifizieren können.

4.2.3 Testausführung 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 und Administrator effizient arbeiten können.

4.2.4 Netzwerktests ausführen

Der Administrator oder Engineer führt die in der Testausführungsplanung spezifizierten Tests auf dem angegebenen System aus. Dazu wird in einer Benutzeroberfläche die Testausführung gestartet oder auf einem Gerät ein stück Code ausgeführt.

4.2.5 Testergebnisse einsehen

Nachdem ein Test durchgeführt wurde, soll ein Testresultat angezeigt werden. In den Resultaten soll ersichtlich sein, welcher Test 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.

4.2.6 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.

4.3 Use Case Diagramm

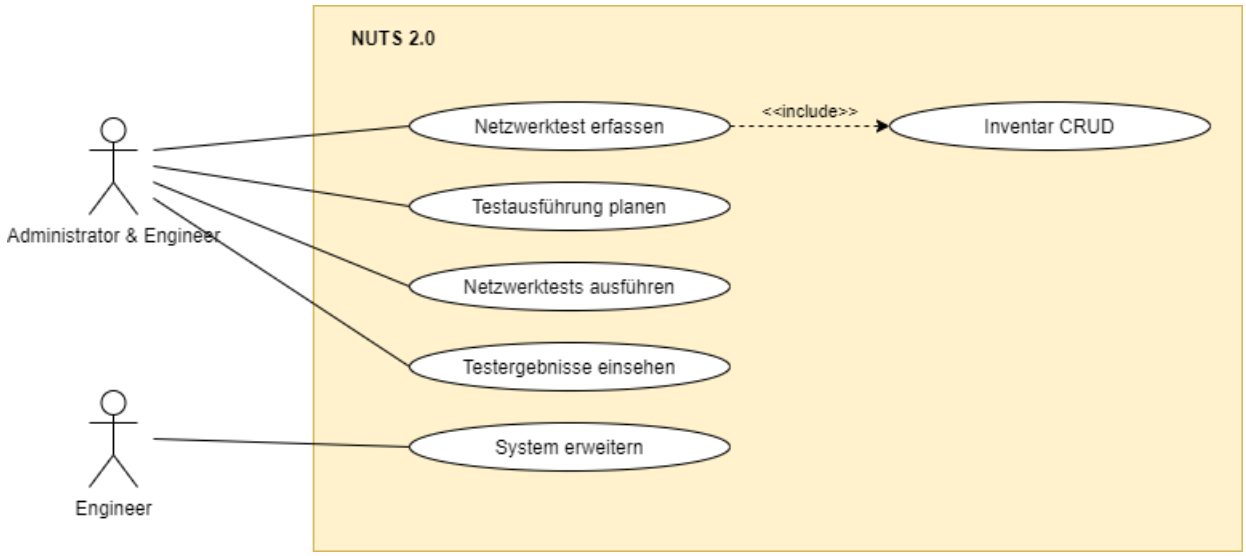

Abbildung 1: Use Case Diagramm

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 betrachtet. Die jeweiligen Aspekte werden in ihren Unterkapiteln genauer beschrieben. Es werden mögliche Szenarien beschrieben, die in der Erstellung oder dem Betrieb der Software auftreten können und bei der Architektur in betracht gezogen werden.

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:

5.1.1 Analysierbarkeit

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

5.1.2 Modifizierbarkeit

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

5.1.3 Stabilität

Wahrscheinlichkeit, dass mit Änderungen unerwartete Nebenwirkungen auftreten.

5.1.4 Testbarkeit

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

5.1.5 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 8: Szenario: Neue Schnittstelle

5.2 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 9: Szenario: Schnelle Fehlerlokalisierung

5.3 Benutzbarkeit

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

5.3.1 Verständlichkeit

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

5.3.2 Erlernbarkeit

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

5.3.3 Bedienbarkeit

Aufwand für den Benutzer, die Anwendung zu bedienen.

5.3.4 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 10: Szenario: Einfachheit der Definitionssprache

5.3.5 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 11: Szenario: Hinweis auf Fehleingaben

5.4 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:

5.4.1 Zeitverhalten

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

5.4.2 Verbrauchsverhalten

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

5.5 Zuverlässigkeit

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

5.5.1 Reife

Geringe Ausfallhäufigkeit durch Fehlzustände.

5.5.2 Fehlertoleranz

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

5.5.3 Wiederherstellbarkeit

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

5.5.4 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 12: Szenario: Testausführung Netzwerkseitig nicht ausführbar

5.6 Betreibbarkeit

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

5.6.1 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.

5.6.2 Installierbarkeit

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

5.6.3 Übertragbarkeit

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

5.6.4 Austauschbarkeit

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

5.6.5 Koexistenz

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

5.6.6 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 13: Szenario: Einfache Installation auf einem anderen Gerät

5.7 Sicherheit

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

5.7.1 Verschlüsselung von Datenübertragungen

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

5.7.2 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.