Anforderungsanalyse

Studienarbeit FS-2020

12. 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.
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.
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.

2.3 Zusätzliche Akteure im zu entwickelnden System

Diese Akteure müssen zusätzlich zu den im Abschnitt 2.2 beschriebenen Entitäten 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.

2.4 Herausforderungen

Das Testen von Netzwerken ist ein komplexes Unterfangen. Die Netzwerkkonfiguration 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-Reiview, 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.

Wie sich diese Anforderungen in einem Netzwerk-Testsystem umsetzen lassen, wird im Abschnitt TODO add reference behandelt.

3 Detailbeschreibung Akteure

3.1 Devices

Für einen Test sind ein, oder mehrere Devices notwendig. Devices sind Geräte im Netzwerk wie z.B. Router und Switches. Um einen Ping abzusetzen braucht es einen Sender und einen Empfänger. Um diese Tests durchführen zu können muss der Benutzer diese Devices hinzufügen, bearbeiten und löschen können.

3.1.1 Devices erfassen

Es bestehen mehrere Möglichkeiten ein Device zu erfassen. Dies wurde in mehrere Stufen unterteilt:

Stufe	Beschreibung
Stufe Manuell	Die Devices werden vom Benutzer manuell in einem File hinzugefügt. Der Benutzer muss hierbei alle Devices eines Netzwerks manuell in ein File übertragen. Jedes Device wird in einem File gespeichert und diese Files werden in einem Repository gespeichert damit, falls das Netzwerk sich einmal ändert, die Devices einach erweitert/angepasst werden können. Das Programm liest danach alle erfassten Devices ein.
Stufe Formular	Der Benutzer kann mit Hilfe eines Formulares der Benutzeroberfläche 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 muss nur noch die Devices erfassen.
Stufe Automatisiert	Die Devices werden automatisch aus dem Netzwerk ausgelesen und hinzugefügt. Der Benutzer muss keine weiteren manuellen Device Eingaben betätigen. You know Mgic and Stuff.

3.1.2 Device Informationen

Je speziefischer die Tests des Programmes sind desto mehr Informationen eines einzelnen Devices werden benötigt. Die Informationen der Devices werden in Kategorien unterteilt: Basic , Advanced , Eskalation. Mit diesen Kategorien wird spezifiziert, wie weit fortgeschritten die Tests sind, die man ausführen möchte.

Property	Beschreibung
Name und Passwort	Diese Informationen werden benötigt, um sich bei einem Device (z.B. Router) anzumelden um dort die Befehle für die Tests auszuführen.
IP Adresse	Die Ip Adresse eines Devices wird benötigt um zum Beispiel einen Ping Test zu machen.
Seriennummer	Spanning Tree bei Switches als deciding Faktor
OSPF Neighbors	OSPF
wiiteri eskalation	wiiteri eskalation

TODO complete Table

3.1.3 Device Gruppen

Die Devices können in Gruppen unterteilt werden. Somit kann man zum Beispiel alle Devices in einem Gebäude in die Gruppe: 'Gebäude 1' schieben. Wenn der Benutzer später einen Test erstellt kann er so direkt alle Devices des Gebäudes auswählen.

3.2 Tests

Dies sind die Tests, die vom Benutzer definiert werden und danach auf dem Netzwerk ausgeführt werden.

3.2.1 Test Kategorien/Typen

Die Tests werden nach Kategorien unterschieden (wie zum Beispiel Connectivity). Diese Kategorien werden auf verschiedene Protokolle und Funktionalitäten unterteilt, um es dem Benutzer möglichst zu vereinfachen. Ein wichtiger Punkt der Test Kategorien ist, dass Benutzer in Zukunft möglichst einfach weitere Kategorien/Testarten hinzufügen können.

3.2.2 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. In diesem File muss der Benutzer die Testart, die beteiligten Devices und weitere Optionen des Testes festlegen.
Stufe Formular	Das Programm stellt ein Formular zur Verfügung, welches die ganze Testerfassung Formular vereinfacht. Der Benutzer muss sich hierbei den Test nur noch über die Benutzeroberfläche zusammenklicken. Das Programm speichert danach den Test automatisch in einem File.

3.2.3 Ergebniss erfassen

Bei gewissen Tests wird ein speziefisches Ergebnis erwartet. Dieses Ergebnis kann der Benutzer bei der Testerfassung eingeben, falls der ausgewählte Testtyp ein Ergebnis verlangt.

3.2.4 Testreihenfolge

Die Tests, die ein Benutzer erfasst können mit Hilfe von Tags in Gruppen unterteilt werden. Mithilfe dieser Gruppen kann der Benutzer später die Reihenfolge der Testdurchführung bestimmen.

3.2.5 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 greift danach über die Netzwerkschnittstelle auf die Devices zu und führt die Tests in der angegebenen Reihenfolge durch.

3.3 Netzwerkschnitstelle

Die Netzwerkschnittstelle ist so gelöst, dass mit minimalen Aufwand jede beliebeige Technologie verwendet werden kann, um eine Connection zu den Devices aufzubauen.

3.4 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 eine Repository gespeichert.

4 Use Cases

4.1 Personas

Person	Beschreibung	Technisches Wissen
Net-Admin	Der Netzwerk-Administrator hat die Verantwortung über das ganze Netzwerk	Er sollte in der Lage sein, Python-Code zu interpretieren und allenfalls zu erweitern.
Net-Engineer	Der Netzwerk-Engineer ist die Person, welche das Netwerk betreibt, er setzt Änderungen und Erweiterungen um und betreibt im Fehlerfall Troubleshooting.	Der Netzwerk-Engineer sollte fundierte Python-Kenntnisse haben und in der Lage sein, das Programm bei Bedarf zu verändern.
Net-Techniker	Der Netzwerk-Techniker unterstützt den Netzwerk-Engineer bei der Wartung und dem Betrieb des Netzwerks.	Der Netzwerk-Techniker hat, wenn überhaupt, nur geringe Kenntnisse über Python. Er soll, ohne Code zu schreiben, dazu in der Lage sein, das Programm auszuführen.

4.2 Use Cases Brief

4.2.1 Tests CRUD

Ein User kann Netzwerktests mit der definierten Sprache definieren. Er benötigt dazu Kenntnisse des Netzwerks und grundlegende Erfahrung in YAML.

4.2.2 Device erfassen

Für einen Test sind ein, oder mehrere Devices notwendig. Devices haben Eigenschaften wie Name, IP-Adresse, Device-Typ (Router, Switch,...) und Login Daten. Diese Devices werden in einer eigenen Sektion in der Testdefinition erfasst.

4.2.3 Command erfassen

Sobald man ein Device erfasst hat, möchte man Kommandos auf diesem ausführen. Dies könnten beispielsweise show-Befehle oder andere Befehle zum Senden von Daten sein.

4.2.4 Ergebnis erfassen

Sobald Devices und Commands erfasst wurden, können nun die zu erwartete Ergebnis formuliert werden. Diese sind als Soll-Werte zu interpretieren und werden vom Programm bei der Durchführung mit den Ist-Werten verglichen.

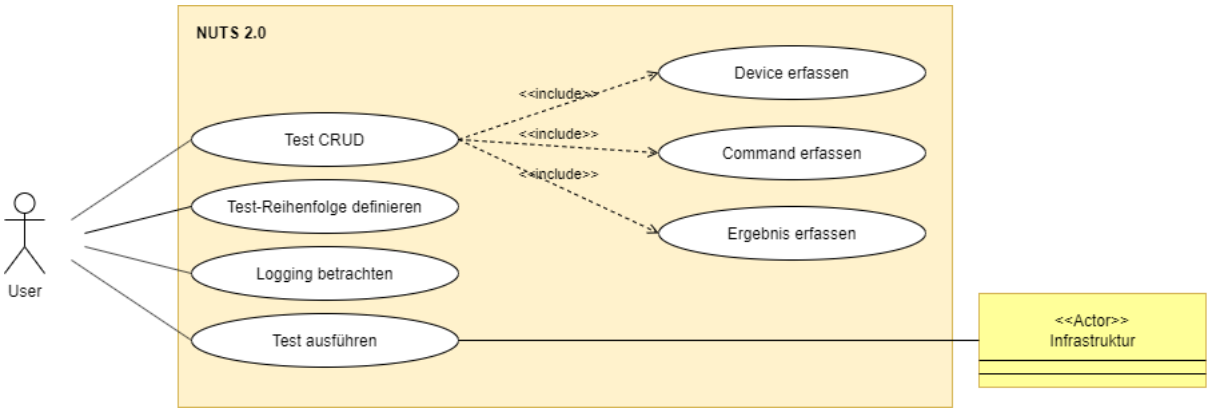
4.2.5 Tests ausführen

Ein fertig formulierter Test kann ausgeführt werden. Dafür wird eine Verbindung zum Netzwerkgerät aufgebaut und das in der Testdefinition spezifizierte Kommando ausgeführt.

4.2.6 Logging/Reports betrachten

Die Auswertung der jeweiligen Durchführung der Tests wird in einem Testreport gespeichert, welchen der User jederzeit einsehen kann, um sich einen Überblich über die Historie vergangener Testdurchführungen zu verschaffen.

4.3 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	Es besteht eine andere Möglichkeit, auf ein Netzwerkgerät über eine Schnittstelle zuzugreifen, die nicht im System integriert und der implementierten Methode zu bevorzugen ist.
Reaktion	Die Software lässt sich von einem Entwickler in weniger als einer Woche um benötigte Komponenten erweitern.
Zielwert	Erweiterungen der Netzwerkschnittstellen sind innerhalb von 40 Personenstunden umsetzbar.

5.1.6 Szenario: Verständlichkeit von generiertem Code

Generierter Code für die Netzwerktests ist leicht verständlich und manuell anpassbar.

Qualitätsziele	Verständlichkeit, Testbarkeit, Modifizierbarkeit
Geschäftsziel(e)	Tester können den generierten Code für die Testsuites und die Testfälle leicht verstehen und ihren eigenen Bedürfnissen anpassen.
Auslöser	Ein Netzwerengineer möchte an der Software Änderungen vornehmen und dafür die Tests anpassen.
Reaktion	Die Testsuites und der Testcode für die Netzwerkeinstellungen werden durch einen Test-Generator in möglichst einfacher Form generiert.
Zielwert	Tester und Entwickler können die generierten Tests in weniger als 30 Minuten verstehen und einfache Anpassungen vornehmen.

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.

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 Testdefinitionen

Die Definitionen von Tests in YAML 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.

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.

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.

5.4.3 Szenario: Schnelle Erzeugung der Netztestdaten

Nach der Erstellung von Testdefinitionen können die Netztests ohne lange Wartezeiten erstellt werden um eine rasche Arbeitsweise zu garantieren.

Qualitätsziele	Performanz, Laufzeitverhalten, Flexibilität, Geschwindigkeit
Geschäftsziel(e)	Die Zeit zwischen der Erstellung und der Durchführung von Tests wird gering gehalten.
Auslöser	Ein Benutzer erstellt Netztests und möchte diese rasch in das System einspeisen.
Reaktion	Tests werdem vom Testgenerator erstellt, so dass die Wartezeiten für Benutzer möglichst gering sind.
Zielwert	Netztests werden in weniger als 10 Sekunden generiert.

5.4.4 Szenario: Optimierte Durchführung von Tests

Die Tests werden möglichst parallel abgearbeitet und nur dann seriell durchgeführt, wenn eine asynchrone Ausführung zu Störungen im Netzwerk führen würden.

Qualitätsziele	Effizienz, geringe Störung im zu testenden Netzwerk, Robustheit
Geschäftsziel(e)	Die Durchführung von Netzwerktests führt nicht zu Performanzeinbussen im Netzwerk.
Auslöser	Es werden mehrere parallelisierbare (z.B. Ping) und mehrere performanzstörende (z.B. Traffic-Test) Tests definiert.
Reaktion	Parallelisierbare Tests werden asynchron durchgeführt. Alle Tests, die das Netzwerk stören können werden nacheinander abgearbeitet.
Zielwert	Es entstehen maximal 30% Performanzeinbussen im Netzwerk während die Tests durchgeführt werden.

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.

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.

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

Zum Zeitpunkt der Erstellung dieses Dokuments ist noch nicht klar, wie mit Passwörtern sicher umgegangen werden soll. Die Usability setzt voraus, dass der Testdurchführer nicht für jeden Test einzeln ein Passwort eingeben muss, Security-verhaltensweisen verbieten aber das Abspeichern von Passwörtern als lesbaren Text.