Architektur

Studienarbeit FS-2020

31. 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

Zweck

Dieses Dokument beschreibt die Architektur und liefert eine Übersicht über die Entscheidungen zum Design und der Architektur des Projektes.

Änderungsgeschichte

Datum	Version	Änderung	Autor
24.03.2018	1.0	Initial Setup	Janik Schlatter
26.03.2020	1.0	Designentscheidungen	Mike Schmid

Inhaltsverzeichnis

1 Designentscheidungen

1.1 Architekturentscheidungen (AE's)

1.1.1 AE 1: Anwendung von Patterns für die Testerstellung

- Im Kontext der Erweiterbarkeit des zu entwickelnden Systems,
- damit weitere Netzwerktest einfach hinzugefügt werden können,
- wurde entschieden, dass für die Testerstellung das Strategy und Factory-Method Pattern angewandt wird,
- um die Instanzierungslogik von den einzelnen Testklassen zu entkoppeln.
- Dabei wird akzeptiert, dass die Implementation des Systems alles in allem komplexer, und die Erstellung umfangreicher wird.

1.1.2 AE 2: Anwendung von Patterns für die Kommunikationsschnittstellen

- Im Kontext der Erweiterbarkeit um weitere Netzwerkschnittstellen,
- damit das Hinzufügen von weiteren Schnittstellen vereinfacht wird,
- wurde entschieden, dass die Auswahl und Instanzierung von Verbindungen über ein Factory Method- und Strategy Pattern realisiert wird.
- um die Verbindungsauswahl vom System zu entkoppeln.
- Es wird dabei akzeptiert, dass das System komplexer wird und die Erstellung mehr Aufwand benötigt.

1.1.3 AE 3: Auswahl der Testdefinitionssprache

- Um die Testdefinitionen in einer möglichst menschenlesbaren Form zu halten,
- und die Verwendbarkeit im zu entwickelnden System zu gewährleisten,
- haben wir entschieden, die Testdefinitionen in YAML zu verfassen,
- um ein Format zu verwenden, dass auch von Netzwerkleuten verwendet wird.
- Andere Technologien, wie JSON, XML usw. werden dabei voraussichtlich wegeglassen,
- auch wenn diese in spezifischen Anwendungen wie die Speicherung von Daten geeignet wären.

1.1.4 AE 4: Weglassen einer Grafischen Benutzeroberfläche (GUI)

- Im Rahmen der zeitlichen Beschränkung der Arbeit,
- um uns auf die Implementation der Logik zu konzentrieren,
- werden wir kein GUI entwickeln.
- Stattdessen wird, wo nötig, eine Kommandozeileninteraktion implementiert.
- Es wird akzeptiert, dass die Benutzerfreundlichkeit des zu entwickelnden Systems dadurch eingeschränkt wird.

1.1.5 AE 5: Verwendung von Nornir

- Für die Kommunikation mit verschiedenen Netzwerken,
- um die Implementation möglichst zu vereinfachen,
- wurde entschieden, dass das System das Nornir-Automations-Framework verwendet wird,
- weil Nornir komplett in Python geschrieben ist und sehr gut mit Python-Code interagiert
- und eine umfangreiche Sammlung von nützlichen Funktionen bietet, mit denen man Netzwerktests ausführen kann.

1.1.6 AE 6: Verzicht auf eine Datenbank

- Um Daten persistent zu speichern,
- damit sie vom System wieder abgerufen werden können,
- werden die Daten in form von Key-Value Paaren im YAML-Format abgelegt,
- und nicht in einer Datenbank,
- wobei ein Datenbanksystem für die zentralisierte Speicherung möglicherweise besser geeignet wäre.

1.1.7 AE 7: Deployment als Python-Skript

- Im Kontext des Deployments, der Auslieferung der Software,
- um das Programm auf möglichst vielen Plattformen einfach einsetzen zu können,
- wird das System in Form eines Python-Scripts ausgeliefert,
- und nicht als ausführbare Datei im .exe Format,
- da Python auf den meisten Linux-Basierten Systemen und einigen modernen Windows-Systemen bereits vorinstalliert ist,
- womit sich die Installation/Ausführung des Programms stark vereinfacht.
- Die Folge, dass auf einigen Geräten für die Programmausführung zuerst Python installiert werden muss, wird akzeptiert.

1.1.8 AE 8: Umgang mit Passwörtern

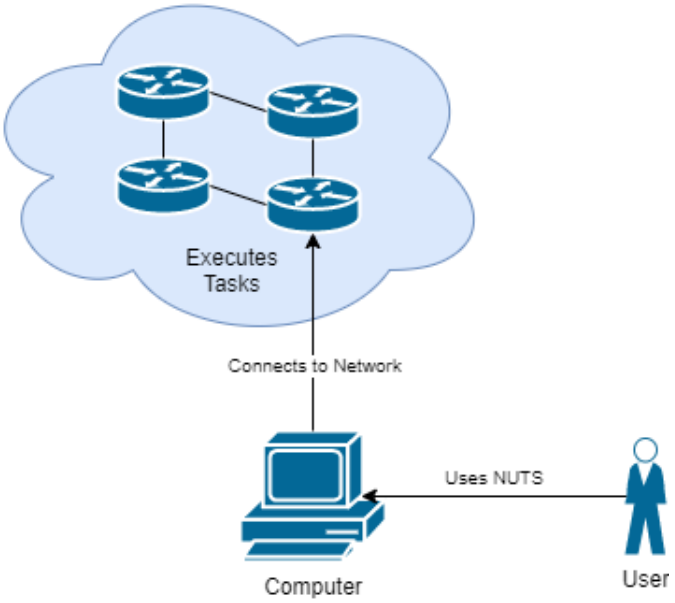
- Im Hinblick auf die Sicherheit des Systems,
- um die Verwendung des Programms möglichst einfach zu halten,
- werden Passwörter für die verschiedenen Netzwerkgeräte als Plaintext gespeichert,
- und nicht in verschlüsselter Form.
- Der sichere Umgang mit den Konfigurationsdaten ist somit in der Verantwortung des Benutzers.

1.2 Umsetzungsentscheidungen

In erster Linie wird das Framework entwickelt, welches es ermöglicht, automatisiert Netzwerktests durchzuführen. Es werden vorerst nur Tests in das System integriert, welche für das Testen der Software benötigt werden. Weitere Tests werden hinzugefügt, wenn im Projekt dafür die Zeit reicht. Zu Beginn wird die Verbindungsschnittstelle Netmiko umgesetzt und das Fundament für die Erweiterung um weitere Schnittstellen gelegt. Die automatisierte Geräteerfassung in ein Inventar wird in Zusammenarbeit mit dem INS, dem Institut für Netzwerklösungen, umgesetzt.

2 Systemübersicht

Die Systemübersicht gibt einen Überblick über die verschiedenen Komponenten des Systems. Nachfolgend sind die einzelnen Komponenten detaillierter beschrieben.



2.1 Computer

Dies entspricht unserem Client. Auf dem Computer läuft das zu entwickelnde System, welches aus einem Python Programm besteht und mittels Nornir mit einem Netzwerk kommuniziert und Tests darauf ausführt.

2.2 Netzwerk

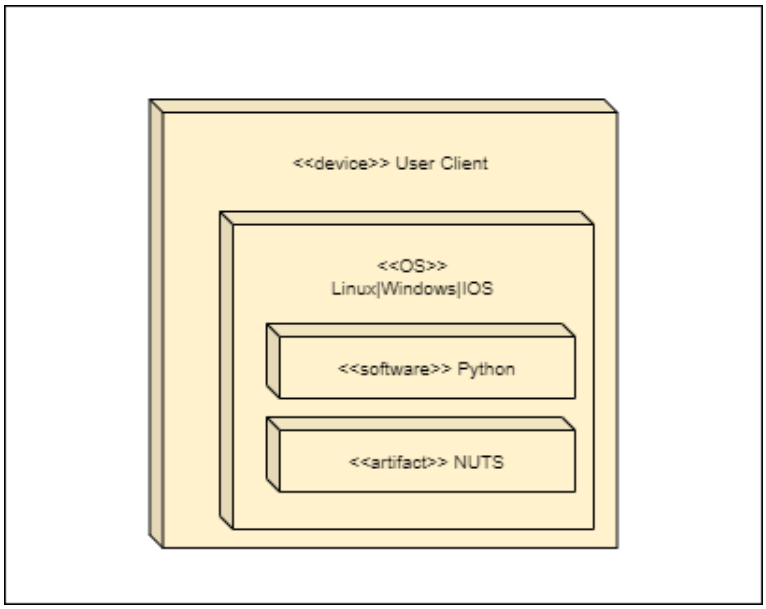
Dies ist das Netzwerk, auf dem die automatisierten Tests ausgeführt werden sollen. Es besteht aus mehreren Netzwerkknoten, z.B. Switches, Router und Clients. Das Netzwerk wird über eine lokale Schnittstelle oder über das Internet angesteuert.

2.3 Datenablage

Die Datenablage befindet sich auf dem Computer und kann bei Bedarf über ein Verwaltungstool mit anderen Clients geteilt werden. Darin befinden sich sämtliche für das Testsystem benötigte Daten. Diese Daten werden in Form von YAML-Files als Key-Value Store angelegt.

3 Deployment

3.1 Deploymentdiagramm



3.2 Client

Der User Client kann ein Computer oder ein Server sein, welcher die automatisierten Tests ausführen soll. Darauf läuft ein beliebiges Betriebssystem

3.3 Betriebssystem

Es gibt keine spezifischen Anforderungen an das Betriebssystem, welches auf dem Client installiert ist. Python Code lässt sich auf beliebigen Betriebssystemen ausführen.

3.4 Python Installation

Für die Ausführung von Python-Code muss Python auf dem Client installiert sein.

3.5 NUTS2.0

Das zu entwickelnde Programm, welches auf dem Client installiert wird. Es verbindet sich mit einem Netzwerk und führt vorher definierte Tests darauf aus.

4 Ausbauszenarien

4.1 Szenario 1: Erweiterung um weitere Neztwerktests

Die Erweiterung des Systems um weitere Tests wird durch die Architektur so einfach wie möglich gehalten. Es müssen lediglich die Testfactory angepasst und neue Testklassen erstellt werden, um neue Tests durch das System zu ermöglichen. Die Tests können danach in der Testdefinition angelegt werden.

4.2 Szenario 2: Erweiterung um Netzwerkschnittstellen

Durch den Einsatz des Factory- und Strategy-Pattern wird die Erweiterung um neue Schnittstellen vereinfacht. Die Factory muss um die neue Schnittstelle erweitert und die Schnittstelle als Klasse implementiert werden. Danach sollte sich die Schnittstelle über die Testdefinition aufrufen lassen.

4.3 Szenario 3: Erweiterung um eine Graphische Benutzeroberfläche (GUI)

Das System wird mit einem Fokus auf die Funktionalität entwickelt. Dabei wird die Usability durch die Verwendung eines GUI vorerst vernachlässigt. Die Erfassung von Netzwerkgeräten, das Erfassen von Tests und die Orchestrierung der Tests liessen sich alle über eine Benutzeroberfläche ansteuern. Nach der Einschätzung der Autoren entspricht die GUI-Erweiterung ungefähr drei bis sechs Wochen Arbeit. Es ist darauf zu achten, die Prinzipien der nutzerzentrierten Entwicklung (User Centered Design) anzuwenden und regelmässig Usability-Tests durchzuführen.

4.4 Szenario 4: Anbindung einer Datenbank

Eine Datenbank würde dem System diverse Möglichkeiten für die effizientere Datenhaltung ermöglichen:

- Geräte liessen sich in einer Graphdatenbank mit den jeweiligen Verbindungen als Relationen speichern, was eine intuitive Darstellung und effiziente Abfragen ermöglicht.
- Durch die Verwendung einer Datenbank liessen sich das Inventar und die Testdefinitionen einfacher von mehreren Clients verwenden und untereinander austauschen.
- Passwörter von Devices, welche nach der aktuellen Architektur in Plaintext gespeichert werden, wären in einer Datenbank besser gesichert.

Auch hier ist mit Aufwand von drei bis vier Wochen zu rechnen. Ausserdem wird für die Haltung einer Datenbank Hardware, beispielsweise ein DB-Server, benötigt.