Domainanalyse

Studienarbeit FS-2020

24. 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

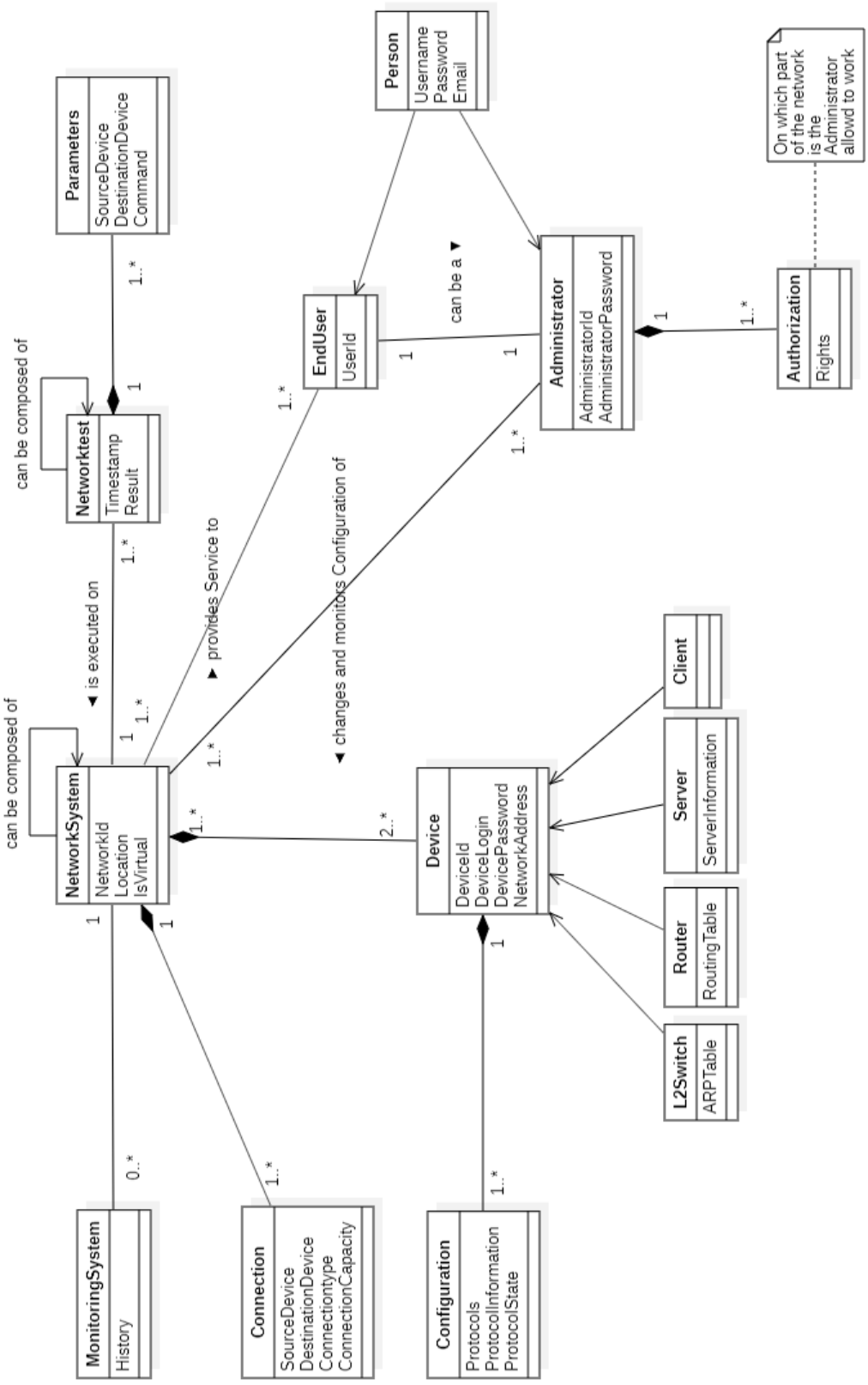
Dieses Dokument soll einen Überblick über die Problemdomäne erlauben und die Konzepte des Domänenmodells erklären. Daraus wird das Klassendiagramm abgeleitet und für die Systemabläufe ein Systemsequenzdiagramm erstellt.

Änderungsgeschichte

Datum	Version	Änderung	Autor
10.03.2020	1.0	Initial Setup	Janik Schlatter
19.03.2020	1.0	DomainModell hinzugefügt	Janik Schlatter
19.03.2020	1.0	Prosa verfasst	Mike Schmid
24.03.2020	1.0	Sequenzdiagramme hinzugefügt	Janik Schlatter

Inhaltsverzeichnis

1 Domänenmodell



1.1 Prosa

Ein Netzwerk (Network System) setzt sich aus mindestens zwei Geräten (Device) und Verbindungen dazwischen (Connection) zusammen. Es kann auch mehrere Teilnetzwerke in sich vereinen, z.B die beiden Netzwerke aus Haupt- und Nebengebäude ergeben das gesamte Firmennetzwerk. Das Netzwerksystem kann auch in virtueller Form aufgebaut sein, Beispielsweise als Netz von virtuellen Routern auf einem Server. Ein Device kann in die vier Kategorien Switch, Router (Level-3-Switch), Server und Client eingeteilt werden und hat eine oder mehrere Konfigurationen. Beispielsweise kann ein Router das OSPF (Open Shortest Path First) Protokoll und zusätzlich als Fallback statische Routen konfiguriert haben.

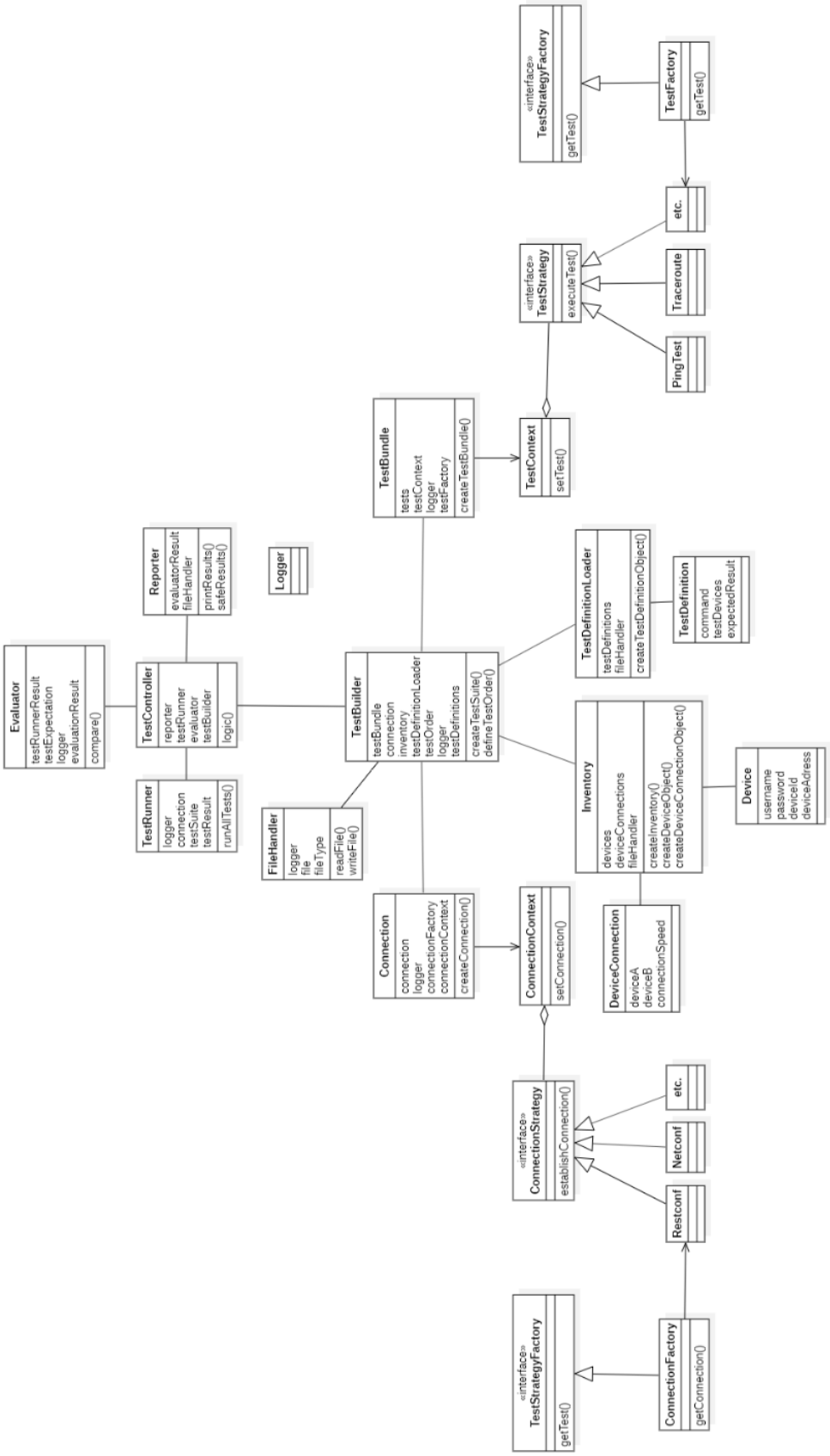
Geräte haben eine Identifikation, ein Gerätelogin und -passwort und eine Addresse innerhalb des Netzwerkes. Die Konfiguration der Geräte setzt sich zusammen aus dem Protokoll, dessen Informationen und dem Zustand, in dem das Gerät mit dem Protokoll gerade ist.

Es ist möglich, dass ein Netzwerksystem ein Monitoring System hat, das die Zustände zur Laufzeit überwacht, z.B. Welche Verbindungen gerade aktiv sind und wieviele Client auf dem Netzwerksystem angemeldet sind. Das Monitoring speichert auch vergangene Daten in einer Historie, so dass vergangene Zustände mit dem aktuellen Zustand verglichen werden können.

Netzwerktests, wie sie von Netzwerk-Technikern auf dem Netzwerk ausgeführt werden, haben Parameter und können aus weiteren Tests zusammengesetzt sein. Ein umfangreicher Systemtest kann aus mehreren Ping-Tests bestehen. Ein Netzwerktest umfasst einen Zeitstempel, um die genaue Durchführungszeit ermitteln zu können und hat ein Resultat, üblicherweise bestanden oder durchgefallen. Die Parameter eines Netzwerktests sind üblicherweise ein Befehl (Ping, Traceroute etc.) und ein Source- und/oder Destination-Device.

Auf dem Netzwerksystem arbeiten verschiedene Personen. Jede Person benötigt einen Usernamen, ein Passwort und eine E-Mail-Addresse, um sich gegenüber dem Neztwerk zu authentifizieren. Diese können in User und Administratoren eingeteilt werden. Usern werden Services vom Netzwerk wie Internet oder Serverzugriff angeboten, und sie haben eine UserID, mit der sie das Netzwerk erkennt. Administratoren können die Konfiguration des Netzwerks einsehen und verändern. Zusätzlich zum regulären Login haben sie einen Administratorzugriff, der aus einer ID und einem Admin-Passwort besteht. Dazu benötigen sie eine Authorisierung, um auf dem Netzwerksystem zu arbeiten.

2 Klassendiagramm



2.1 Beschreibungen

2.1.1 TestController

Der TestController ist das Kernstück des Programms. Er beinhaltet die Main-Methode und steuert den Ablauf des Programms. Vom TestController werden der TestRunner, der TestBuilder, der Evaluator und der Reporter instanziert und er ist zuständig für die Kommunikation zwischen diesen Komponenten.

Komponenten	Beschreibung
reporter	Referenz auf das Reporter-Objekt
testRunner	Referenz auf das TestRunner-Objekt
evaluator	Referenz auf das Evaluator-Objekt
testBuilder	Referenz auf das TestBuilder-Objekt
logic()	Methode, die für die Programmausführung sämtliche referenzierten Komponenten instanziert und deren Funktionalität in der korrekten Reihenfolge ausführt.

2.1.2 TestRunner

Der TestRunner führt die ihm vom Controller mitgeteilten Tests gemäss der, in der TestSuite angegebenen, Parameter aus. Die zu verwendende Netzwerkschnittstelle (Restconf, Netconf, SSH etc.) wird ihm ebenfalls vom Controller mitgeteilt. Die Resultate der Netzwerktests gibt er dem TestController in Form von Rückgabewerten zurück.

Komponenten	Beschreibung
logger	Referenz auf das Logger-Objekt
connection	Referenz auf das Connection-Objekt
testSuite	Collection von Testspezifikationen, welche Tests sollen auf welchen Netzwerkkomponenten in welcher Reihenfoge ausgeführt werden.
testResult	Resultat der Netzwerktests die dem Controller nach abarbeiten aller Tests zurückgegeben werden.
runAllTests()	Methode, die die in der Testsuite spezifizierten Tests auf der in der Connection spezifizierten Netzwerkschnittstelle ausführt.

2.1.3 Reporter

Der Reporter schreibt die Testergebnisse der fertig ausgeführten Tests auf die Konsole/Benutzeroberfläche und erstellt ein Testprotokoll, welches er über den FileHandler im Directory abspeichert.

Komponenten	Beschreibung
evaluationResult	Die Ergebnisse des Evaluators, wie der Soll- Ist-Vergleich der Tests abgelaufen ist.
fileHandler	Referenz auf das FileHandler-Objekt.
printResults()	Methode die die Testergebnisse auf der Konsole ausgibt.
safeResults()	Methode, die über den FileHandler die Testergebnisse in einem File im Directory abspeichert.

2.1.4 Evaluator

Der Evaluator vergleicht die ihm vom Controller mitgeteilten Testresultate mit den Testerwartungswerten und evaluiert, ob die Tests bestanden sind oder nicht.

Komponenten	Beschreibung
testRunnerResult	Ergebnisse der auf dem Netzwerk vom Runner ausgeführten Tests.
testExpectation	Zu erwartende Ergebnisse für einen spezifischen Test der Testsuite.
logger	Referenz auf das Logger-Objekt.
evaluationResult	Resultat des Soll-Ist-Vergleichs, welches dem Controller zurückgegeben wird.
compare()	Methode, die testRunnerResult mit testExpectation vergleicht und entscheidet, ob der Test erfolgreich war, oder gescheitert ist.

2.1.5 FileHandler

Der FileHandler ist eine Utility-Klasse, die für das Einlesen und Schreiben von Daten aus dem Programm in das Directory zuständig ist.

Komponenten	Beschreibung
logger	Referenz auf das Logger-Objekt.
file	Pfad zu dem zu lesenden/schreibenden File.
fileType	Typ des Files, YAML, XML, JSON usw.
readFile()	Methode, die ein vorgegebenes File öffnet und deren Inhalt in das Programm einliest.
writeFile()	Methode, die einen vorgegebenen Text aus dem Programm in ein File im Directory schreibt.

2.1.6 Testbuilder

Der Testbuilder ist für die Zusammenstellung der Tests verantwortlich. Er instanziert einen TestDefinitionLoader, der aus dem TestDefinitionsDirectory, worin mehrere Files mit TestDefinitionen gespeichert sind, die einzelnen TestDefinitionen ausliest. Dann holt er sich aus dem Inventory die Devices und deren Connections. Die TestDefinitionen attributisiert er mit den, in der TestStrategy definierten Tests, und erstellt mit den Parametern aus dem Inventar das TestBundle. Hier kann vom Benutzer auch die Auswahl der einzelnen Tests, welche durchgeführt werden sollen, sowie die Durchführungsreihenfolge festgelegt werden. Die Connection spezifiziert die konkrete Netzwerkschnittstelle, über welche die Netzwerktests vom Runner dann durchgeführt werden sollen.

Komponenten	Beschreibung
testBundle	Collection von Tests mit Parametern für die Ausführung als Referent auf ein TestBundle-Objekt.
connection	Referenz auf das Connection-Objekt.
inventory	Referenz auf das Inventory-Objekt.
testDefinitionLoader	Referenz auf ein TestDefinitionLoader-Objekt.
testOrder	Definition, in welcher Reihenfolge die Tests ausgeführt werden sollen.
logger	Referenz auf das Logger-Objekt.
testDefinitions	Collection mit Referenzen auf TestDefinition-Objekte.
createTestSuite()	Methode, die aus den destDefinitionen, dem Inventar und der Testreihenfolge eine TestSuite erstellt.
defineTestOrder()	Methode, die die Testreihenfolge vom Softwareuser einstellen lassen kann.

2.1.7 TestBundle

Das TestBundle ist dafür zuständig, gemäss der TestDefinitionen die Tests zusammenzustellen. Dazu wird ein TestContext und eine TestFactory instanziert und damit die Tests ausgewählt und instanziert.

Komponenten	Beschreibung
tests	Collection von Tests, die von der TestFactory instanziert wurden und dem TestBuilder als Rückgabewert zurückgegeben wird.
testContext	Referenz auf das TestContext-Objekt.
logger	Referenz auf das Logger-Objekt.
testFactory	Referenz auf das TestFactory objekt.
createTestBundle()	Methode, die aus den TestDefinitionen über eine TestFactory die konkreten Tests auswählt und in einer Collection zusammenfasst.

2.1.8 TestContext

Der TestContext wird benötigt, um unter Anwendung des Strategy-Pattern die Auswahl der Tests durchzuführen. Die Methode setText() ruft dabei die Factory auf und instanziert die konkreten Tests.

2.1.9 TestStrategy

Das TestStrategy-Interface dient als Basis für die konkreten Implementationen der Tests. Das Interface gibt den Test-Classes dafür die benötigte Funktionalität vor, die wiederum von den konkreten Tests implementiert werden müssen. Die ExecuteTest() Methode ist ein Beispiel für eine solche Funktionalität.

2.1.10 TestFactory

Die TestFactory ist die Anwendung des Factory Method Pattern, welches den Tests erlaubt, instanziert zu werden, ohne dass sich diese selbst um die Instanzierungslogik kümmern müssen. Die Factory entscheidet dabei, welche Tests für welche Definitionen instanziert werden müssen und instanziert diese konkreten Tests mit der getTest() Methode.

2.1.11 TestDefinitionLoader

Der TestDefinitionLoader ist dafür zuständeg, die TestDefinitionen aus dem Directory zu laden und als einzelne TestDefinitionen zu instanzieren.

Komponenten	Beschreibung
testDefinitions	Collection von TestDefinitionen, die dem TestBuilder als Rückgabewerte zurückgegeben werden.
fileHandler	Referenz auf das FileHandler-Objekt.
createTestDefinitionObject	Methode, die die TestDefinitionen aus dem FileSystem ausliest und TestDefinition-Objekte für jede Definition instanziert.

2.1.12 Inventory

Das Inventory ist diejenige Klasse, die im Programm die einzelnen Geräte und deren Verbindungen untereinander verwaltet. Sie liest dazu aus dem FileSystem die spezifizierten Devices und Device-Connections ein und instanziert Klassen, um diese als Collection dem TestBuilder zurückzugeben.

Komponenten	Beschreibung
devices	Collection der eingelesenen Geräte.
deviceConnection	Collection der Geräteverbindungen.
fileHandler	Referenz auf das FileHandler-Objekt.
createInventory()	Methode, die das Inventar erstellt.
createDeviceObject	Methode, die aus dem FileSystem die einzelnen Devices ausliest und für jedes ein Device-Objekt erstellt.
createDeviceConnectionObject	Methode, die aus dem FileSystem die Geräteverbindungen ausliest und für jede Verbindung ein deviceConnection-Objekt erstellt.

2.1.13 Connection

Die Connection-Klasse spezifiziert die Netzwerk-Schnittstelle, die für die Verbindung zwischen dem Programm und des zu Testenden Netzwerks verwendet werden soll. Für die Auswahl und Instanzierung wird eine ConnectionFactory verwendet und die Verbindungen lassen sich mittels einem Strategy-Pattern auswählen.

Komponenten	Beschreibung
connection	Ausgewählte Netzwerkschnittstelle, die dem TestBuilder als Rückgabewert geliefert wird.
logger	Referenz auf das Logger-Objekt.
connectionFactory	Referenz auf das ConnectionFactory-Objekt.
connectionContext	Referenz auf das ConnectionContext-Objekt.
createConnection()	Methode, die aufgrund der Parameter der Devices eine mögliche Netzwerkschnittstelle auswählt.

2.1.14 ConnectionContext

Anwendung des Strategy Pattern für die Netzwerkschnittstelle. Der ConnectionContext hält eine Referenz auf das konkrete Schnittstellen-Objekt.

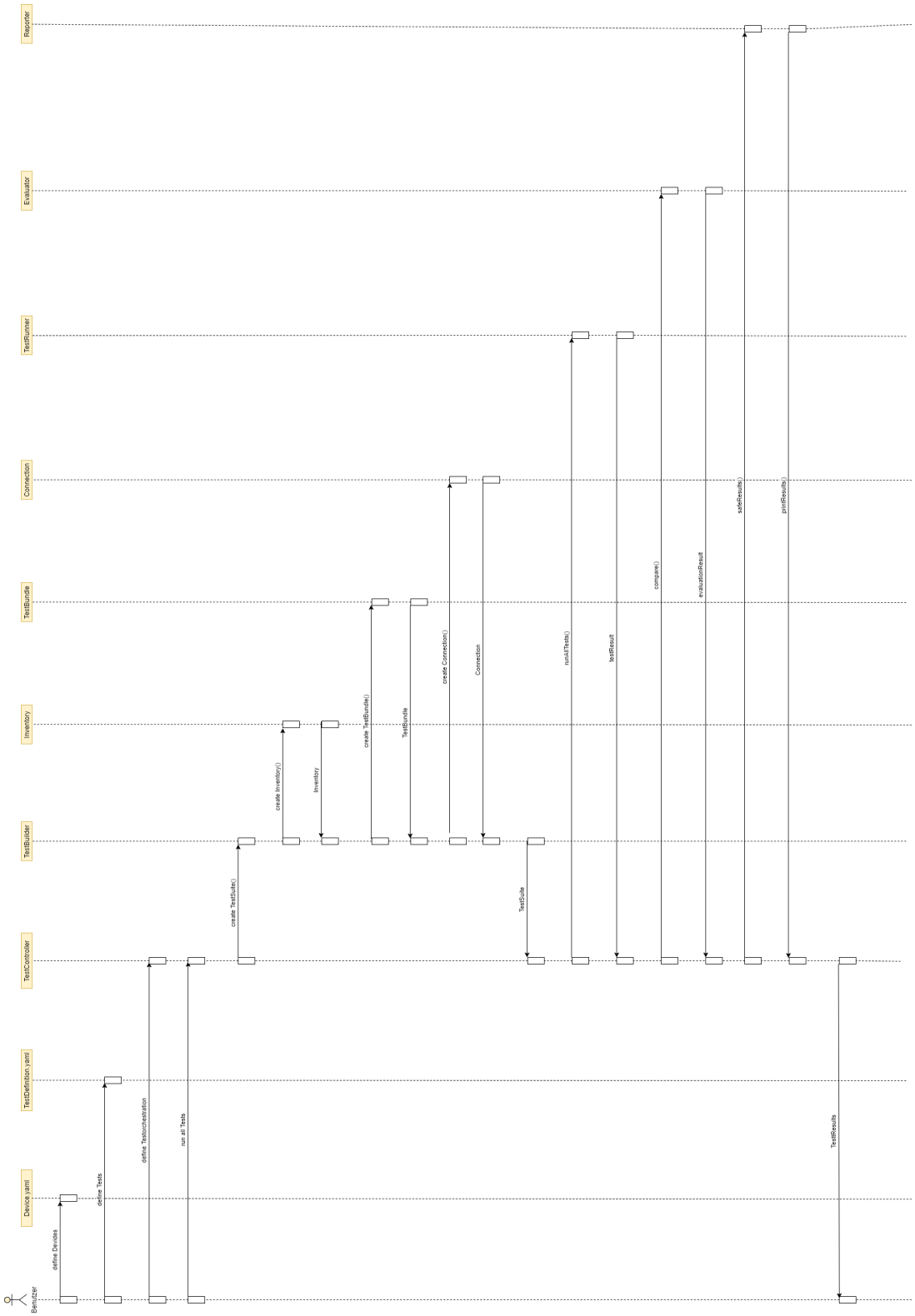
2.1.15 ConnectionFactory

Anwendung des Factory-Pattern auf die Netzwerkschnittstelle. Die Factory ist zuständig für die Instanzierung der konkreten Schnittstelle, mit der das Programm die Netzwerkumgebung testet.

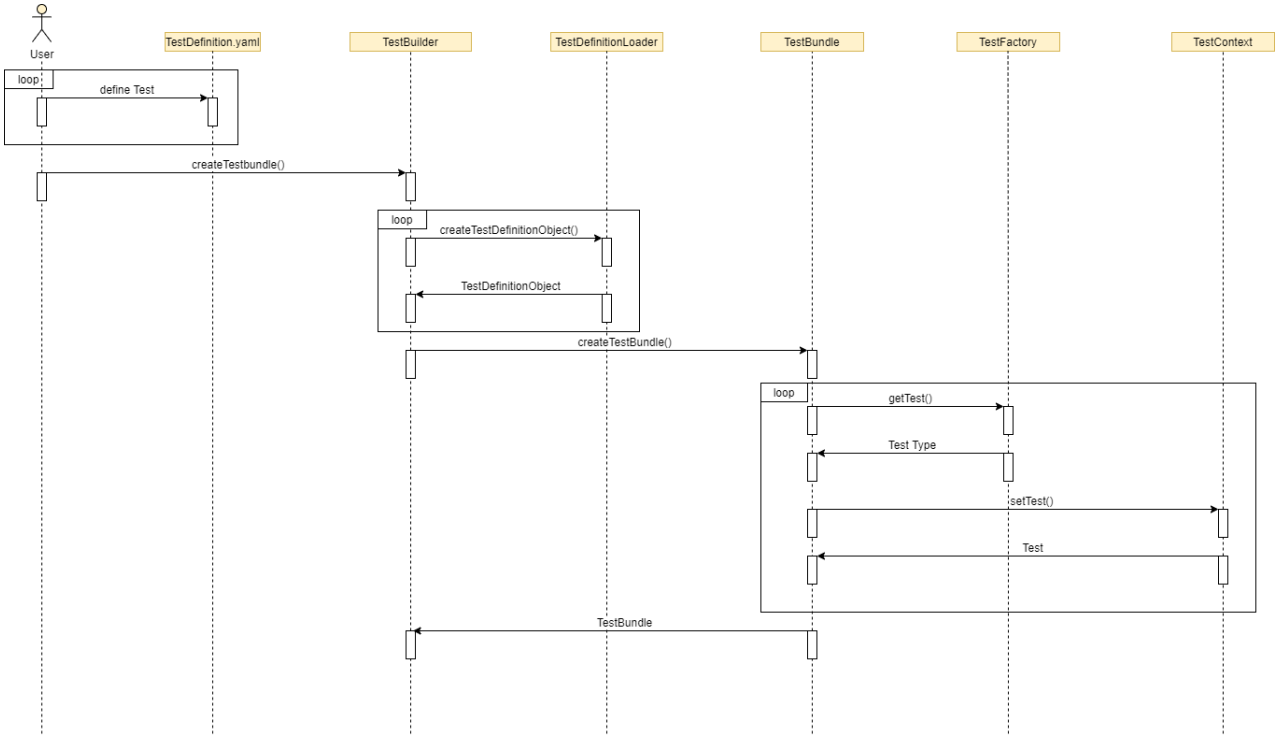
2.1.16 Logger

Der Logger wird verwendet, um wichtiche Informationen zentral zu speichern. Diese Informationen betreffen nur den Systemzustand des zu entwickelnden Systems. Der Logger speichert Fehlermeldungen, Erfolgsbenachrichtigungen und weitere Informationen, die es einem Entwickler erlauben, unerwartetes Verhalten der Software besser zu verstehen.

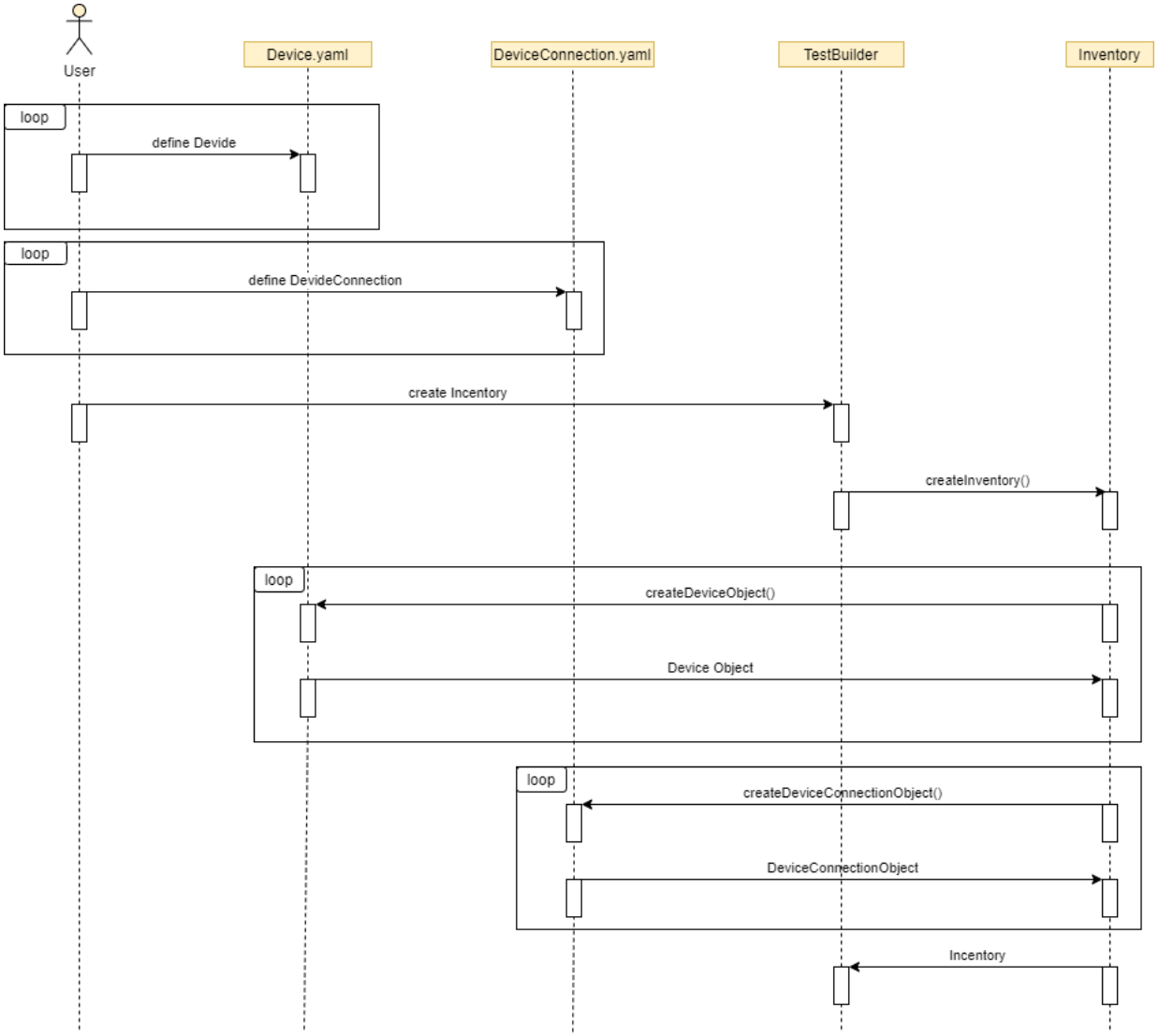
3 Systemsequenzdiagramme



3.1 TestBundle



3.2 Inventar


User
Device.yaml
DeviceConnection.yaml
TestBuilder
Inventory
loop
define Devide
loop
define DevideConnection
create Incentory
createInventory()
loop
createDeviceObject()
Device Object
loop
createDeviceConnectionObject()
DeviceConnectionObject
Incentory

3.3 Connection