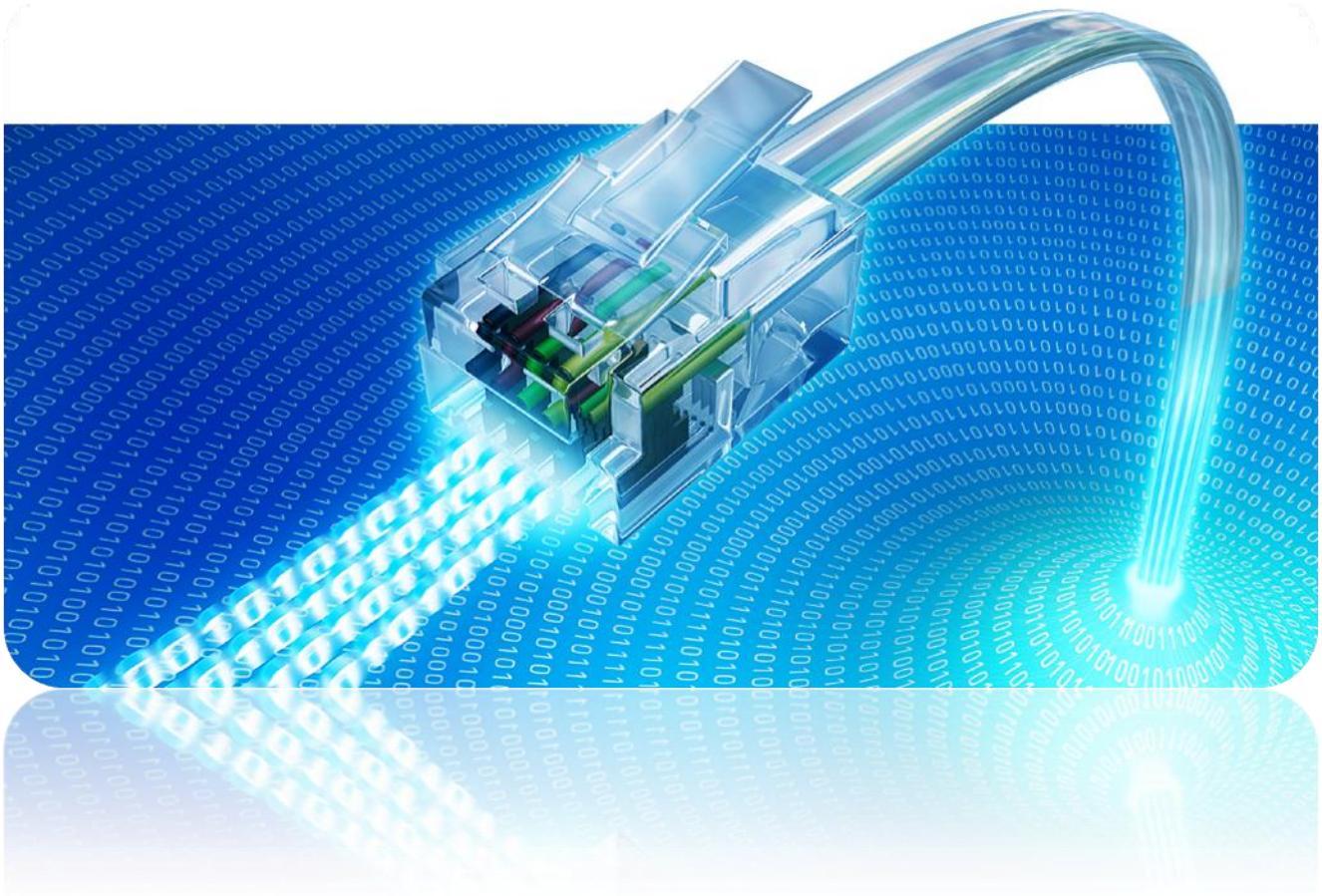
HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

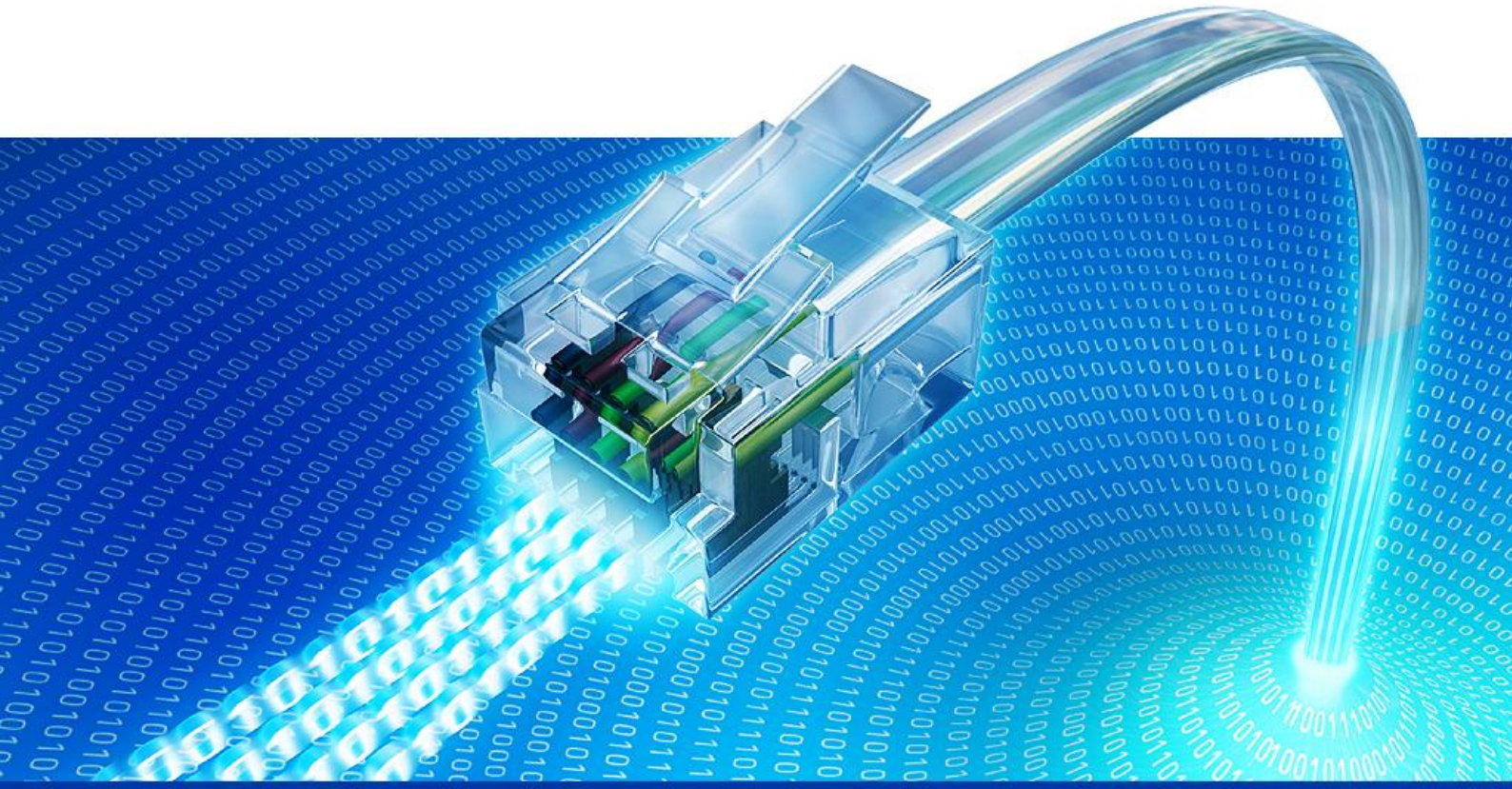
ins institute for networked solutions

Bachelorarbeit FS10

Network Testcase Engine



Autoren:	Oliver Zürcher
	Ymyr Osman
Betreuer:	Prof. Beat Stettler
	Michael Schneider
Experte:	Markus Vögtlin, Clounet AG
Gegenleser:	Andreas Steffen



1. Abstract

Network Testcase Engine

Bachelorarbeit FS 2010

Technische Hochschule Rapperswil

1. Abstract

Das INS hat 2003 für den Flughafen Zürich die Internet Access Plattform MPP entwickelt. Seither hat sich der Kundenstamm auf Hochschulen, Grossfirmen und Service-Provider vergrössert. Auch die Anzahl Clients ist an diesen Hotspots in den letzten Jahren regelrecht explodiert, was zu einem sehr starken Anstieg der Last auf den Portalen geführt hat.
Durch den grossen Zuwachs verkompliziert sich die Durchführung von Tests gegen das MPP. Die Anzahl Clients für ein in der Realität entsprechendes Szenario ist bereits zu gross, um solche Tests von Hand effizient durchführen zu können. Es soll nun ein Applikation zum effizienten Testen des MPPs entwickelt werden.

Da ein Computer alleine nicht mehrere tausend Clients (virtuelle Maschinen) simulieren kann, werden diese auf mehrere Computer verteilt (sogenannte Nodes). Das verteilte System wird über eine Webapplikation gesteuert. Ein Service stellt der Webapplikation eine Schnittstelle zur Steuerung der Nodes und VMs zur Verfügung.

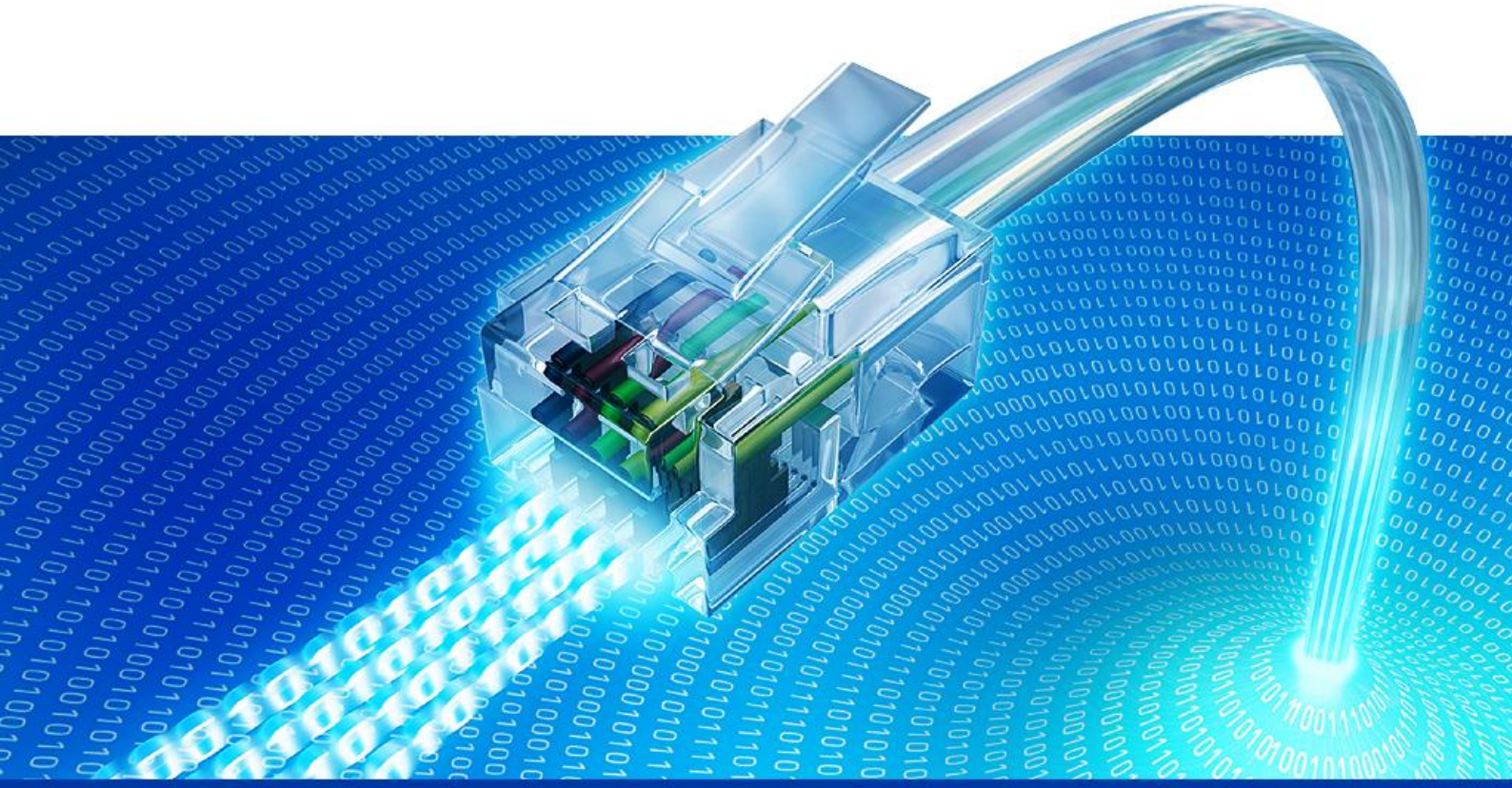
In der Webapplikation können Szenarien definiert werden. Diese legen fest, welche Testcases auf wieviele Clients verteilt und wie oft diese wiederholt werden sollen. Testcases bestehen aus einzelnen Tasks, welche workflowartig zusammengebaut werden können.

Nachdem alle Nodes via Multicast im Netzwerk gefunden worden sind, kann über die Webapplikation ein Szenario gestartet werden. Es werden automatisch genügend virtuelle Maschinen auf die Nodes verteilt, aufgestartet und ein Testcase zugewiesen. Der Fortschritt des Prozesses wird in der Webapplikation angezeigt. Die VMs warten nach dem Aufstarten auf ein weiteres Multicast Start-Paket, bevor sie mit der Ausführung des Testcases beginnen.

Die Ergebnisse der einzelnen Testcases eines Szenariodurchlaufes werden in der Datenbank gespeichert und sind in der Webapplikation einsehbar.

Mit der Task-API können weitere Tasks implementiert werden, welche in der Webapplikation in Testcases verwendet werden können.

Auf den Nodes kommt die aktuelle Linux Distribution Ubuntu 10.4 „Lucid Lynx“ zum Einsatz. Auf den virtuellen Maschinen wird Debian 5.0 „Lenny“ eingesetzt.



2. Management Summary

Network Testcase Engine

Bachelorarbeit FS 2010

Technische Hochschule Rapperswil

1. Ausgangslage

Das INS hat 2003 für den Flughafen Zürich die Internet Access Plattform MPP entwickelt. Seither hat sich der Kundenstamm auf Hochschulen, Grossfirmen und Service-Provider vergrössert. Auch die Anzahl Clients ist an diesen Hotspots in den letzten Jahren regelrecht explodiert, was zu einem sehr starken Anstieg der Last auf den Portalen geführt hat.
Durch den grossen Zuwachs verkompliziert sich die Durchführung von Tests gegen das MPP. Die Anzahl Clients für ein in der Realität entsprechendes Szenario ist bereits zu gross, um solche Tests von Hand effizient durchführen zu können.
In der Semesterarbeit „Captive Portal Load Generator" wurde ein lauffähiger Prototyp entwickelt. Dieser soll nun um folgende Punkte erweitert werden:

- Konfiguration / Verwaltung auf einer Webapplikation, sodass kein fester Client mehr notwendig ist
- Die Clients sollen Testcases analog zu Unit-Tests ausführen
- Testcases können workflowartig in der Webapplikation zusammengestellt werden (aus verschiedenen Tasks)
- API um die Software mit weiteren Tasks zu erweitern
- Webapplikation stateless

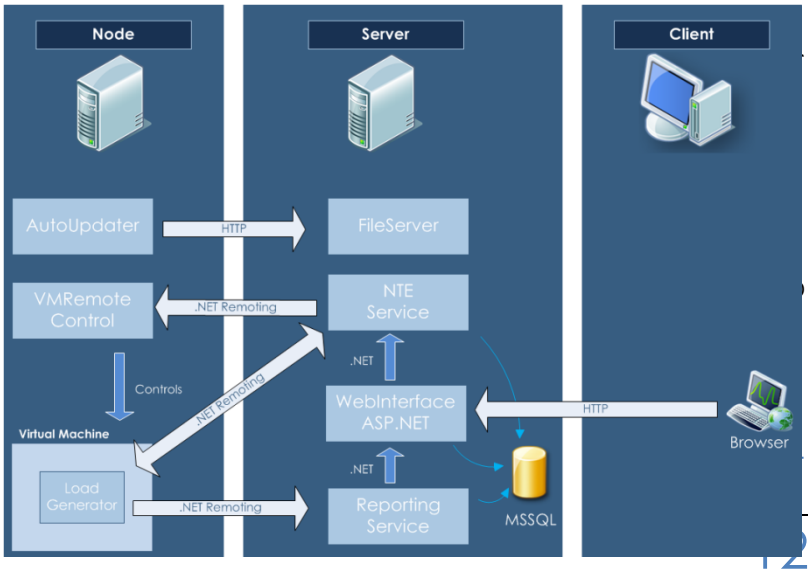
2. Vorgehen / Technologien

In der Vorarbeit erwies sich Kernel-based Virtual Machine (KVM) als die optimale Lösung zur Virtualisation vieler Clients. Deshalb kommt auch bei der Network Testcase Engine KVM zum Einsatz.
In einer ersten Analyse wurden die verschiedenen möglichen Tasks definiert und die Architektur festgelegt.
Als Webserver kommt ein Windows Server 2008 R2 zum Einsatz. Für die persistierung wird ein Microsoft SQL 2008 Server eingesetzt. Als OR-Mapper wird das neue Entity Framework 4 CTP3 verwendet. Die Webseite wird in ASP.NET programmiert, wobei der grösste Teil des Codes in Javascript geschrieben wird. Zum Einsatz kommt hier das Javascript Framework ExtJS. Auf den Linux Rechner kommt .NET 3.5 unter der Mono Runtime zum Einsatz. Für die Interprozesskommunikation (IPC) wird auf das .NET Remoting Framework zugegriffen, welches seit .NET 2.0 Bestandteil des Frameworkes ist und somit auch unter Mono funktioniert.

3. Ergebnisse

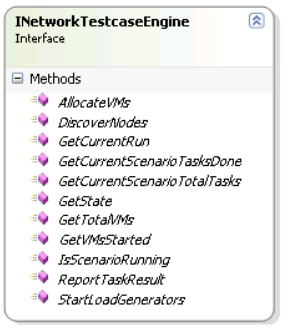
3.1 Architektur

Es können beliebig viele Nodes dem verteilten System hinzugefügt werden, um die Anzahl virtueller Maschinen zu erhöhen. Der Benutzer selbst interagiert über eine Webapplikation mit dem ganzen System. Die Webapplikation wird von einem Service unterstützt, welcher längere Aufgaben entgegen nimmt und verarbeitet.



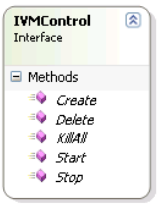
3.2 NTE.Service.exe

Der Service stellt dem Webinterface via .NET Remoting Methoden zur Interaktion mit den Nodes und VMs zur Verfügung. So übernimmt der Service längere Prozesse wie zum Beispiel das Aufstarten mehrere VMs. Diese können nicht innerhalb eines einzelnen Webrequests durchgeführt werden (Webseiten sind kurzlebig).



3.3 NTE.VMRemoteControl.exe

Die Konsolenanwendung VMRemoteControl läuft auf jeder Node, welche seine Hardware für die Testszenarien zur Verfügung stellt. Auf ein „register_request“ Multicast Paket antwortet es mit Informationen von der Hardware, und wie es via .NET Remoting erreichbar ist (IP/Port). Mit diesen Informationen kann der Service virtuelle Maschinen auf der Node verwalten und steuern. Innerhalb wenigen Millisekunden kann eine VM anhand eines Base-Images erstellt werden (Overlay-Image).

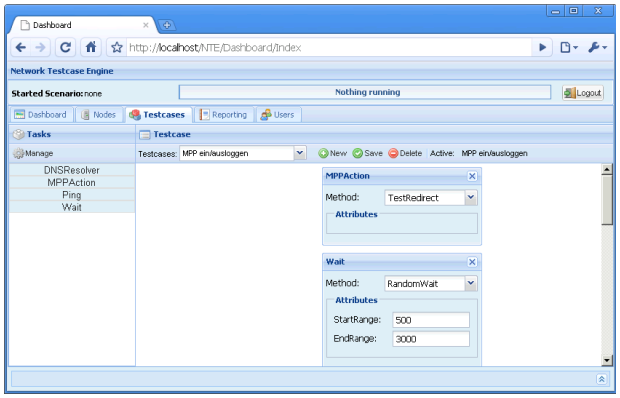


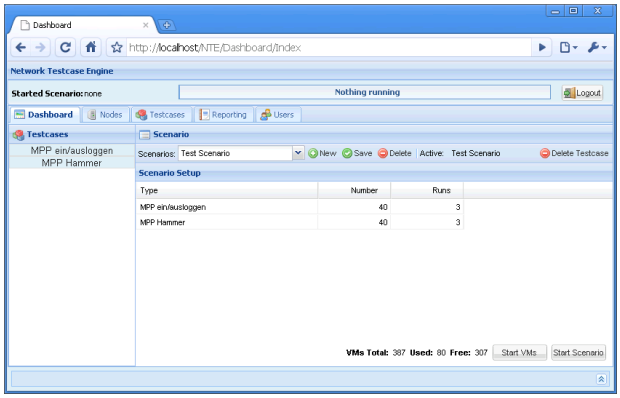
3.4 NTE.LoadGenerator.exe

Die LoadGenerator Anwendung ist im Basis Image der virtuellen Maschinen vorinstalliert und startet nach dem Bootvorgang automatisch auf. Vom NTE.Service wird ein Testcase zugewiesen, welcher nach einem Multicast Start-Paket ausgeführt wird. Ergebnise raportiert die Applikation zurück an den Service, damit die Ergebnisse in der Webapplikation aufbereitet werden können.

3.5 NTE.Webinterface

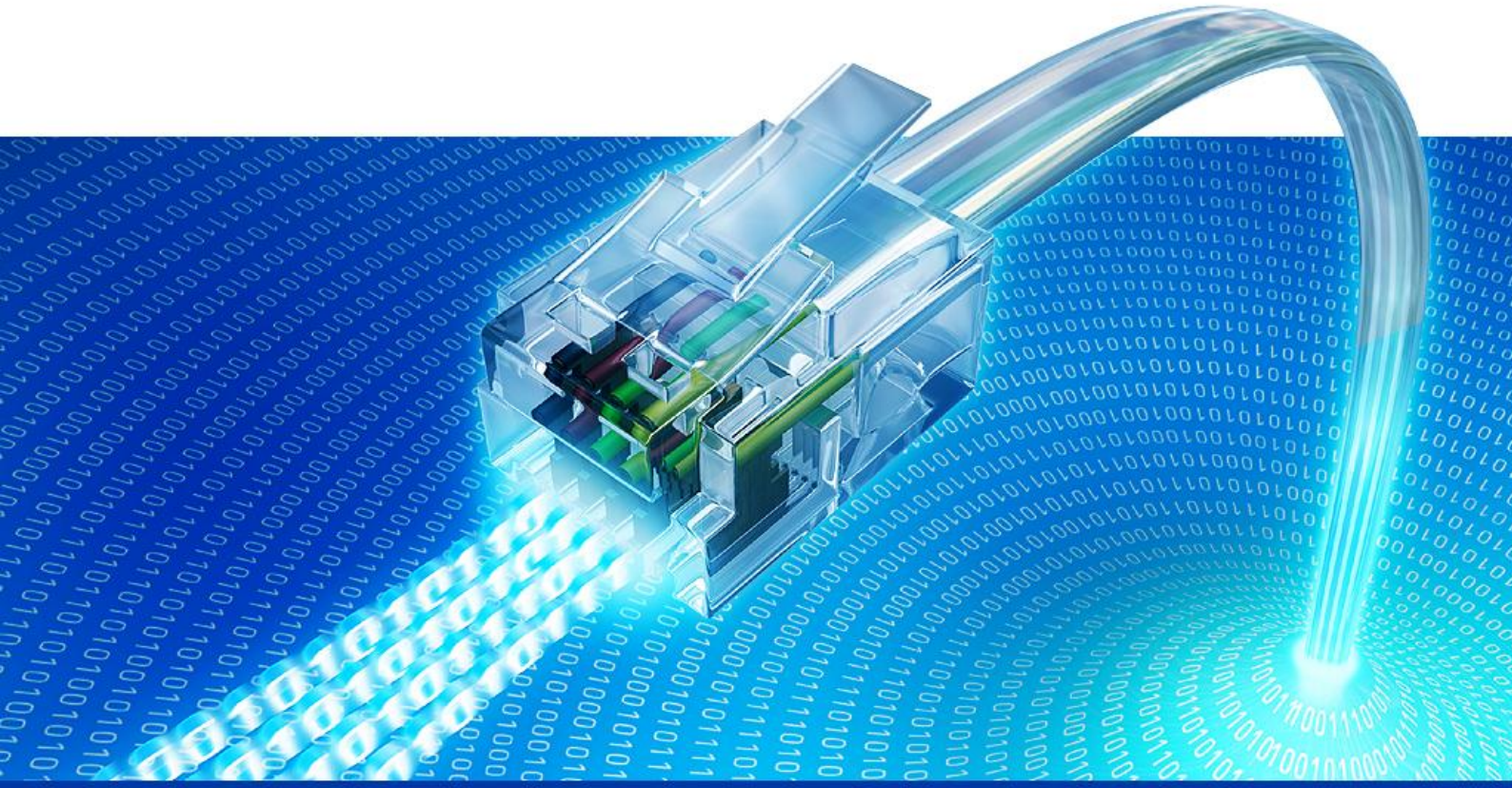
Der Benutzer interagiert mit der Webapplikation. Testcases können erstellt (linkes Bild) und in Szenarien eingebettet werden (rechtes Bild). In Szenarien wird definiert, welche Testcases auf wieviele VMs verteilt und wie oft sie wiederholt werden sollen.





4. Ausblick

Mit der Network Testcase Engine kann das MPP auf eine effiziente Weise getestet werden. Testcases lassen sich mit wenigen Klicks zusammenbauen und anschliessend simultan auf hunderte virtualisierte Clients ausführen.
Durch die Task-API können in Zukunft bei Bedarf weitere Tasks implementiert werden, um die Software zu erweitern.
Beim Captive Portal Load Generator konnte das MPP mit 200 simultanen Clients, welche gleichzeitig einen Request abschickten, direkt abgeschossen werden. So konnte das MPP nur schwer getestet werden. Selbst in der Realität sind 200 Requests in den selben ~10ms relativ unrealistisch. Bei der Network Testcase Engine kann nun ein Random-Wait vor einen Testcase geschaltet werden, der dieses Problem löst. Dadurch starten die einzelnen Testcases zu unterschiedlichen Zeiten und hämmern nicht alle gleichzeitig auf das MPP los.



3. Technischer Bericht

Network Testcase Engine

Bachelorarbeit FS 2010

Technische Hochschule Rapperswil

1. Aufgabenstellung

Das INS hat 2003 für den Flughafen Zürich die Internet Access Plattform MPP entwickelt. Seither hat sich der Kundenstamm auf Hochschulen, Grossfirmen und Service-Provider vergrössert. Auch die Anzahl Clients ist an diesen Hotspots in den letzten Jahren regelrecht explodiert, was zu einem sehr starken Anstieg der Last auf den Portalen geführt hat.
Durch den grossen Zuwachs verkompliziert sich die Durchführung von Tests gegen das MPP. Die Anzahl Clients für ein in der Realität entsprechendes Szenario ist bereits zu gross, um solche Tests von Hand effizient durchführen zu können.
In der Semesterarbeit „Captive Portal Load Generator" wurde ein lauffähiger Prototyp entwickelt. Dieser soll nun um folgende Punkte erweitert werden:

- Konfiguration / Verwaltung auf einer Webapplikation, sodass kein fester Client mehr notwendig ist
- Die Clients sollen Testcases analog zu Unit-Tests ausführen
- Testcases können workflowartig in der Webapplikation zusammengestellt werden (aus verschiedenen Tasks)
- API um die Software mit weiteren Tasks zu erweitern
- Webapplikation stateless

2. Vorgehen

2.1 Analyse der Tasks

Basierend auf den LoadGenerators der Vorarbeit wurden neue Tasks definiert, welche in Testcases eingesetzt werden können. Folgende Tasks wurden definiert:

Task		Beschreibung
	TestRedirect	Surft eine Webseite an, und überprüft ob das MPP auf die Landing-Page weiterleitet
	TestLoggedIn	Besucht eine Website und überprüft, ob diese geladen werden kann, oder das MPP auf die Landing-Page umleitet.
	Wait	Wartet eine gewünschte Zeit in Millisekunden.
	RandomWait	Wartet zufällig in einem vorgegebenen Zeitraum
	Ping	Überprüft ob Ping zu einem Host möglich ist
	DNSResolve	Versucht einen Domainnamen aufzulösen
	MPPLogin	Loggt sich beim MPP ein
	MPPLogout	Loggt sich beim MPP aus
	Google	Sucht nach ein paar Begriffen bei Google und besucht die Ergebnisse

2.2 Architektur

Als Architektur wurde ein verteiltes System gewählt, welches um beliebig viele Nodes erweitert werden kann.

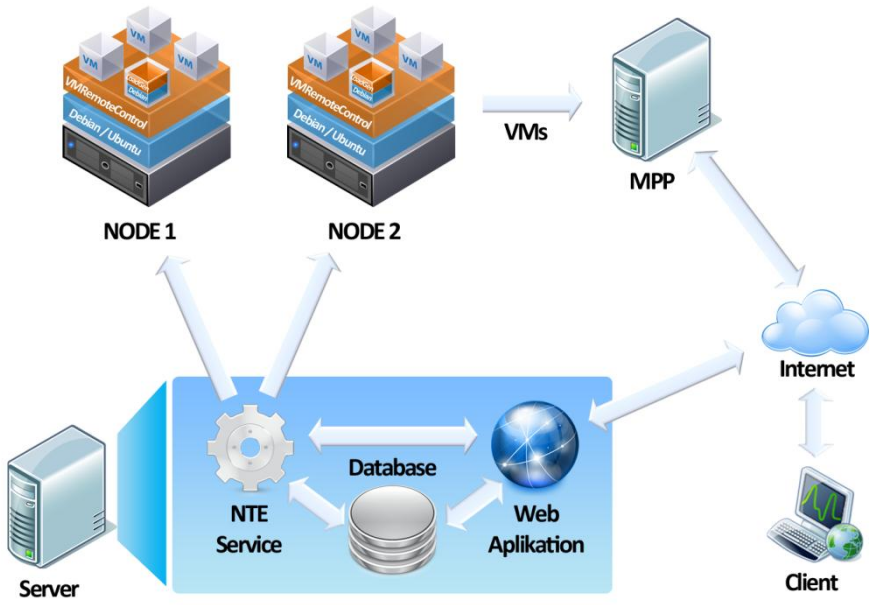


Abbildung 1: Architektur

Anders als beim Captive Portal Load Generator kommuniziert nun ein Service mit den Nodes und VMs. Für die Benutzerinteraktion ist eine Webapplikation zuständig, welche mit dem Service kommuniziert. Als gemeinsame Datenablage wird eine SQL Datenbank verwendet.

2.3 Virtualisierung

Für die Virtualisierung wird KVM eingesetzt. Eine ausführliche Analyse wurde bereits in der Vorarbeit durchgeführt. Auf diesem Ergebnis wird in dieser Bachelorarbeit aufgebaut und deshalb KVM verwendet.
Auf allen Computer (Nodes), welche ihre Hardware für virtualisierte Clients der Network Testcase Engine zur Verfügung stellen, wird ein Basisimage vorinstalliert. Einzelne VMs verwenden ein Overlay-Image davon, welches innert wenigen Millisekunden erstellt werden kann. Die benötigte Software ist im Basisimage bereits vorinstalliert und wird beim Bootvorgang automatisch aufgestartet.
Für jede virtuelle Maschine wird mit TUN/TAP eine Netzwerkkarte mit eigener MAC Adresse simuliert (Das MPP identifiziert Clients auf MAC Ebene).

3. Ergebnis

3.1 Service.exe

Der Service stellt dem Webinterface via .NET Remoting Methoden zur Interaktion mit den Nodes und VMs zur Verfügung. So übernimmt der Service längere Prozesse wie zum Beispiel das aufstarten mehrere VMs. Diese können nicht innerhalb eines einzelnen Webrequests durchgeführt werden (Webseiten sind kurzlebig).

➔ Network Testcase Engine (Kapitel 3.3)

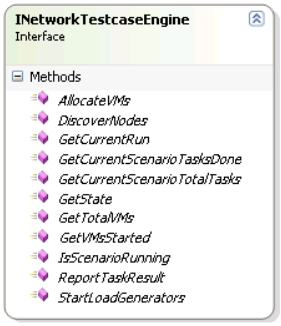


Abbildung 2: Schnittstelle vom Service

3.2 VMRemoteControl

Die Konsolenanwendung VMRemoteControl läuft auf jeder Node, welcher seine Hardware für die Testszenarien zur Verfügung stellt. Auf ein „register_request" Multicast Paket antwortet es mit Informationen von der Hardware, und wie es via .NET Remoting erreichbar ist (IP/Port). Mit diesen Informationen kann der Service virtuelle Maschinen auf der Node verwalten und steuern. Innerhalb wenigen Millisekunden kann eine VM anhand eines Base-Images erstellt werden (Overlay-Image).

➔ KVM Host *n* (Kapitel 3.3)

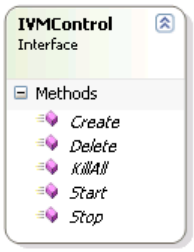


Abbildung 3: Schnittstelle von VMRemoteControl

3.3 Hardware Aufstellung

Folgendes Schema zeigt eine mögliche Hardwareaufstellung auf. Im Rahmen der Bachelorarbeit wurde die Data Storage, der Webserver und die Network Testcase Engine auf dem selben Server installiert.

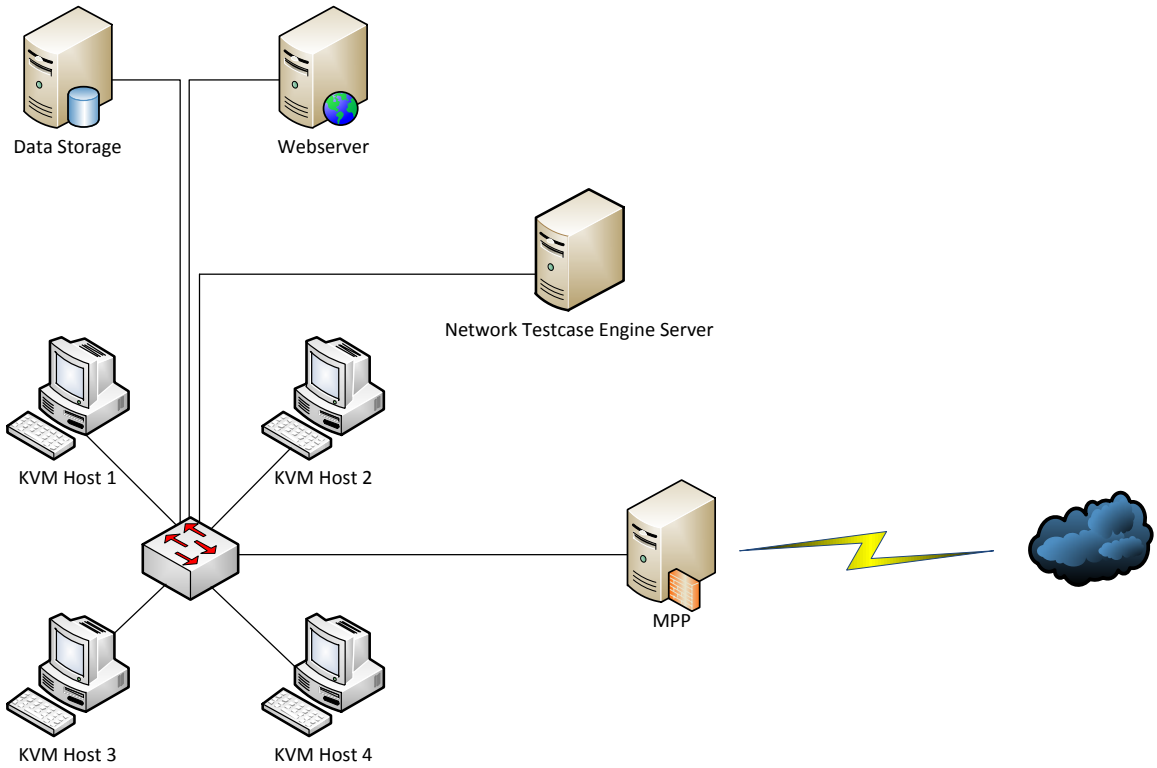


Abbildung 4: Hardwareaufstellung

Wenn nötig können dem System beliebig weitere KVM Hosts hinzugefügt werden. Die Verteilung der VMs übernimmt die Network Testcase Engine automatisch.

3.4 Web Frontend

Die Network Testcase Engine wird über das Web Frontend gesteuert. Testcases können aus einzelnen Tasks per Drag&Drop zusammengestellt und anschliessend in Szenarien eingebettet werden. Beim Start eines Szenarios werden automatisch genügend VMs auf die KVM Hosts verteilt und aufgestartet. Nach dem Startvorgang können die VMs im Webinterface angewiesen werden, ihre Testcases auszuführen. Dazu wird ein Multicast Paket verschickt, um allen VMs das Signal möglichst gleichzeitig zu geben.
Während ein Szenario läuft, können die Logs der virtuellen Maschinen überprüft werden. Im Tab Reports werden die Ergebnisse von Szenariodurchläufen aufgelistet.

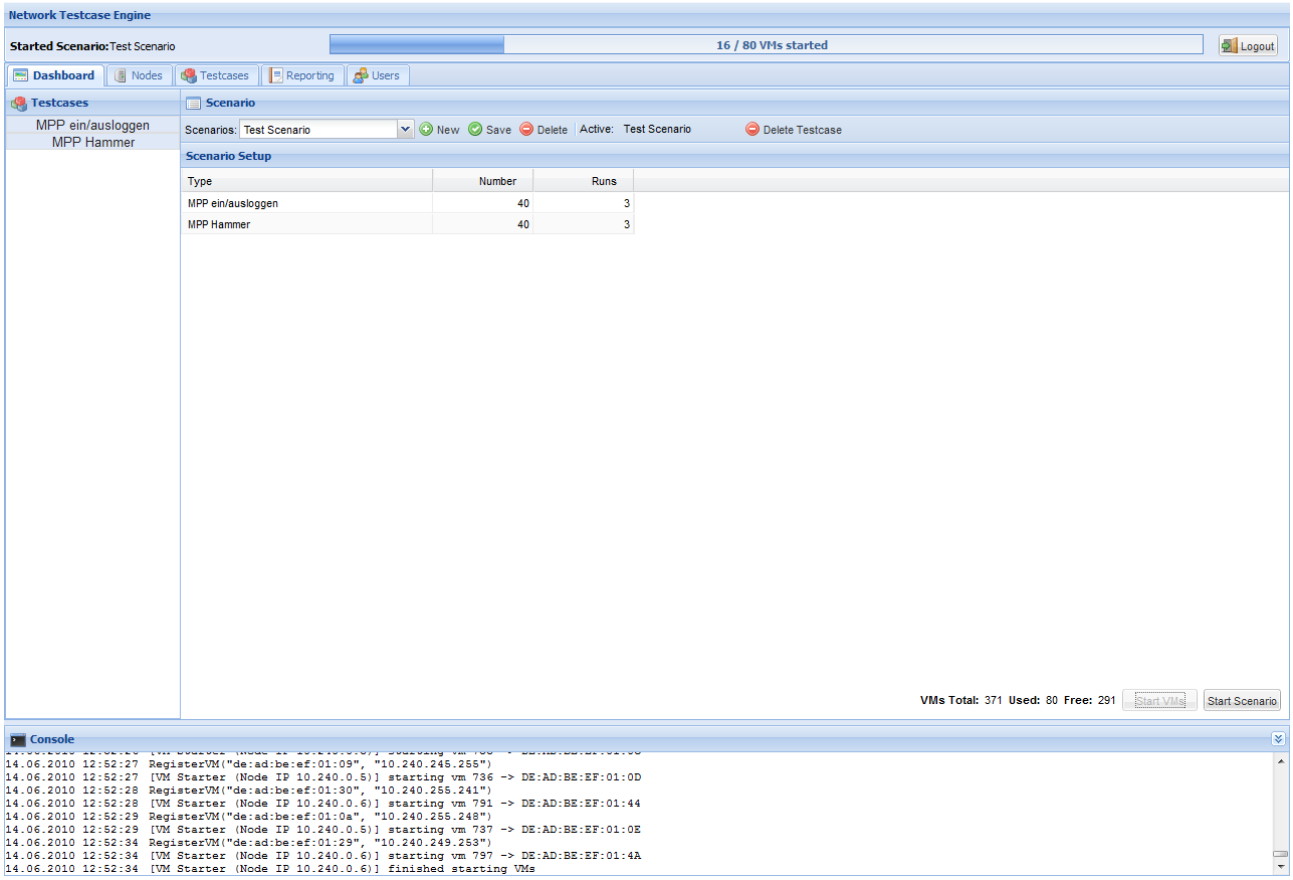


Abbildung 5: Szenario Konfigurator

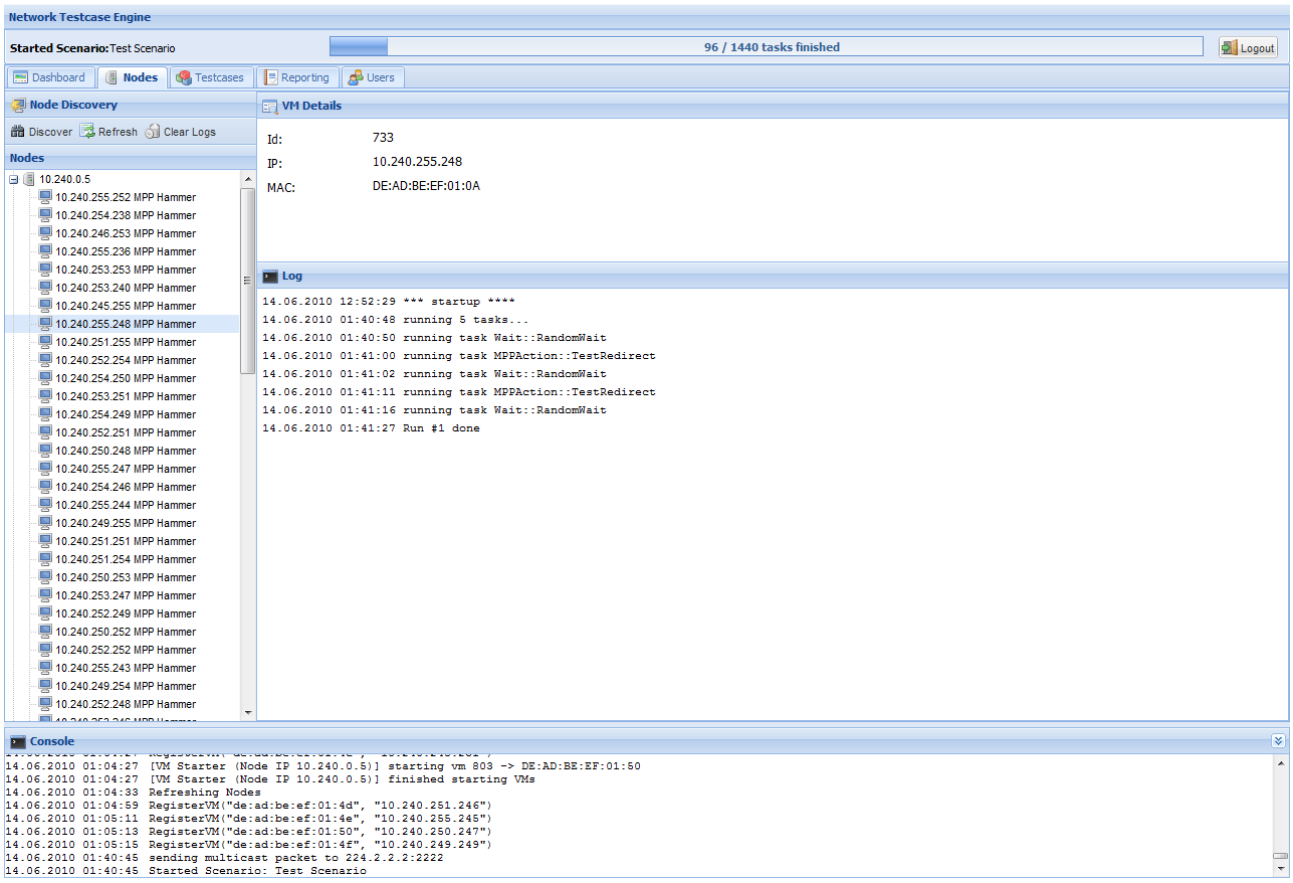


Abbildung 6: Log Ansicht einer VM

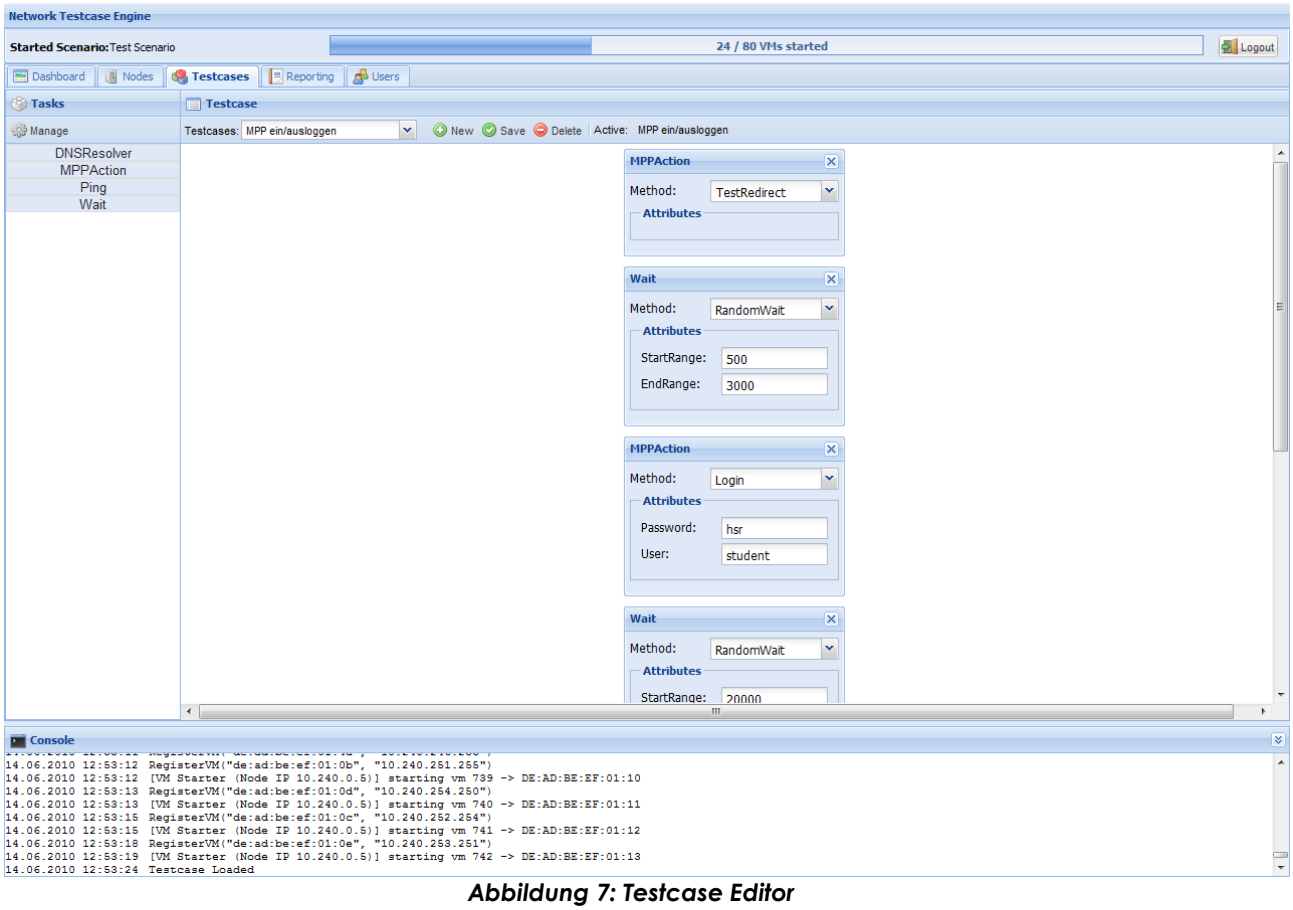


Abbildung 7: Testcase Editor

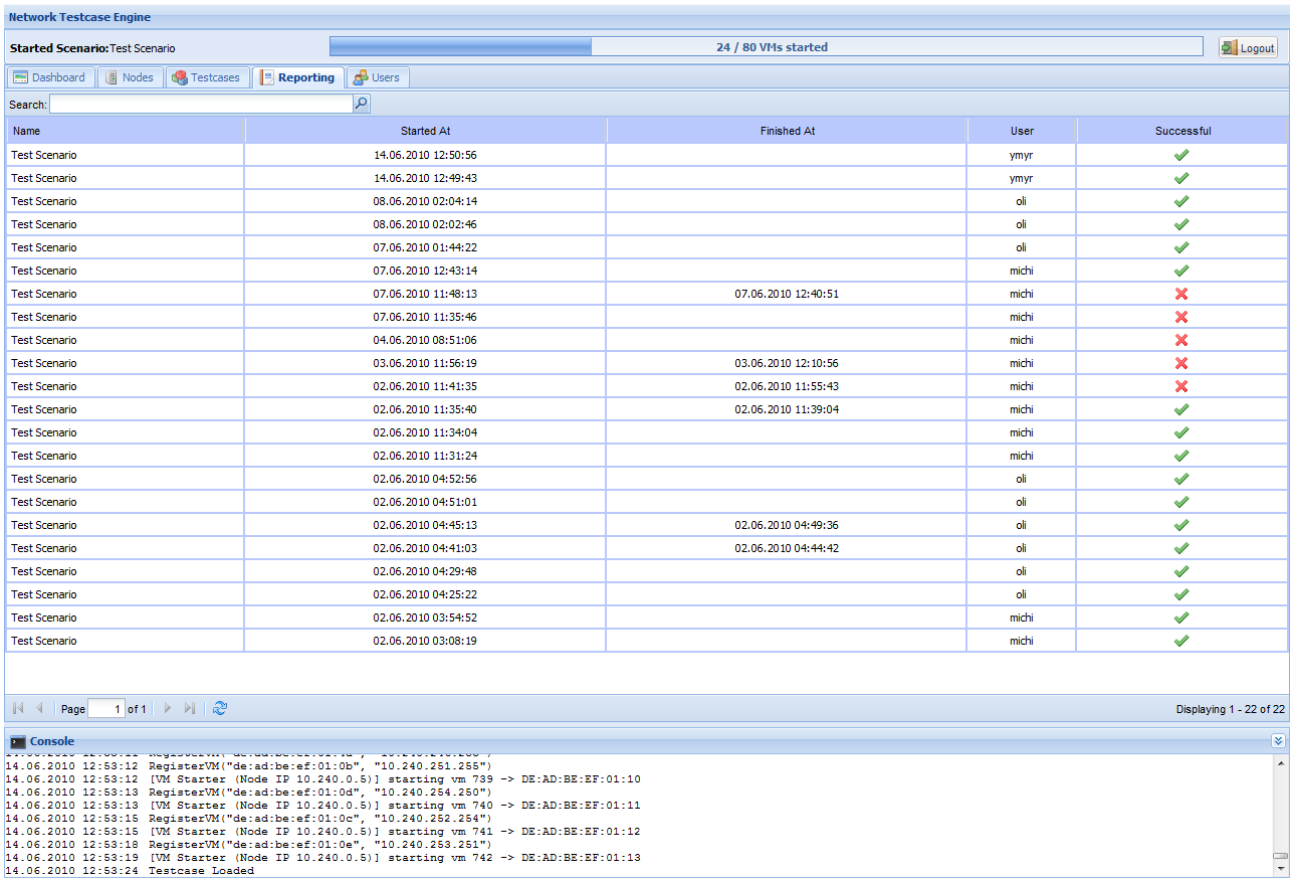


Abbildung 8: Reports Tabelle

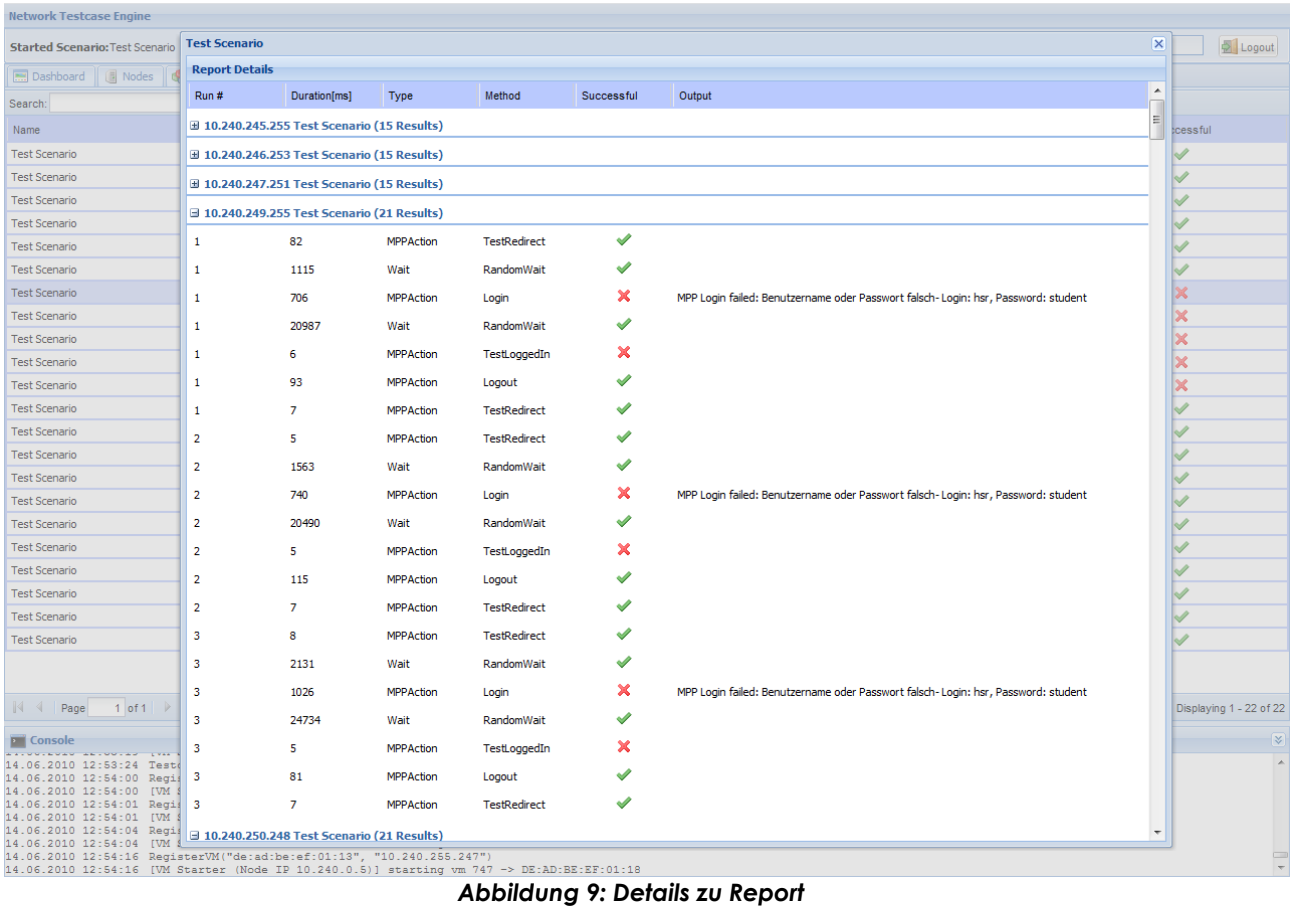


Abbildung 9: Details zu Report

3.5 Task API

Die Network Testcase Engine verfügt über eine Task API, damit neue Tasks schnell und effizient implementiert werden können. Tasks müssen als DLL in der Webapplikation und im Baseimage installiert werden. Sie werden automatisch auf der Webseite im Task Editor erkannt und werden auf den virtuellen Maschinen automatisch geladen. Folgende Schritte sind für die korrekte Implementation eines Tasks notwendig:

1. Klasse die Public ist und von TaskBase (aus NTE.Common.dll) ableitet erstellen
2. Notwendige Properties erstellen und mit `[TaskProperty]` annotieren
3. Public Methode erstellen, welche den Code des Tasks enthält. Diese mit `[TaskMethod]` annotieren, damit die Methode im Task Editor ausgewählt werden kann.
4. Alle Task Properties, welche vom Task verwendet werden, müssen mit `[RequiredProperty("PropertyName")]` an der Methode annotiert werden

3.5.1 Beispiel Taskimplementation

```
namespace NTE.Task.ExampleTask
{
    public class ExampleTask : TaskBase
    {
        [TaskProperty]
        public string Property1 { get; set; }

        [TaskProperty]
        public int Property2 { get; set; }

        [TaskMethod()]
        [RequiredProperty("Property1")]
        [RequiredProperty("Property2")]
        public bool PingIP()
        {
            return true; // do some stuff
        }
    }
}
```

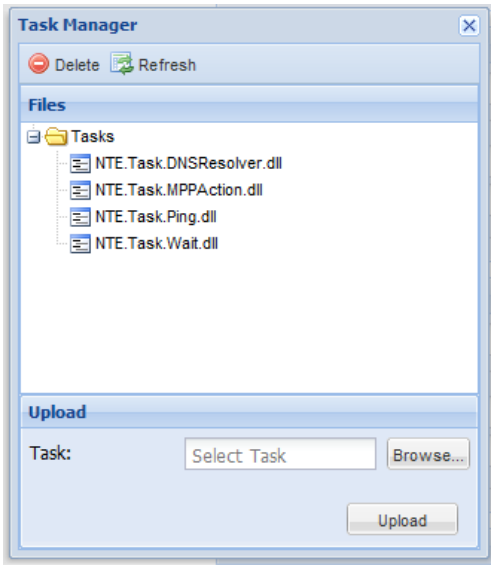
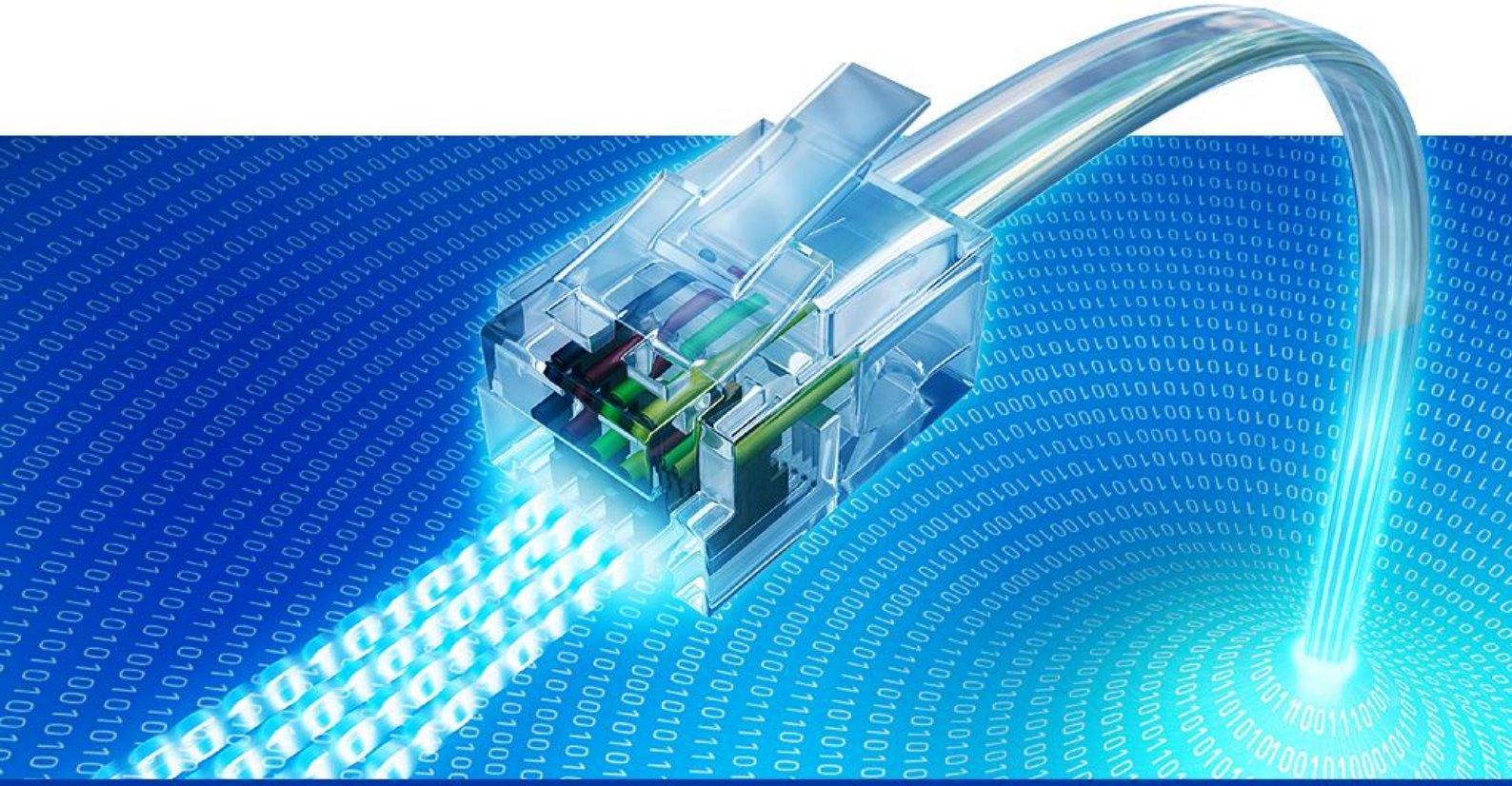


Abbildung 10: Task DLL in der Webapplikation registrieren



4. Persönliche Berichte

Network Testcase Engine

Bachelorarbeit FS 2010

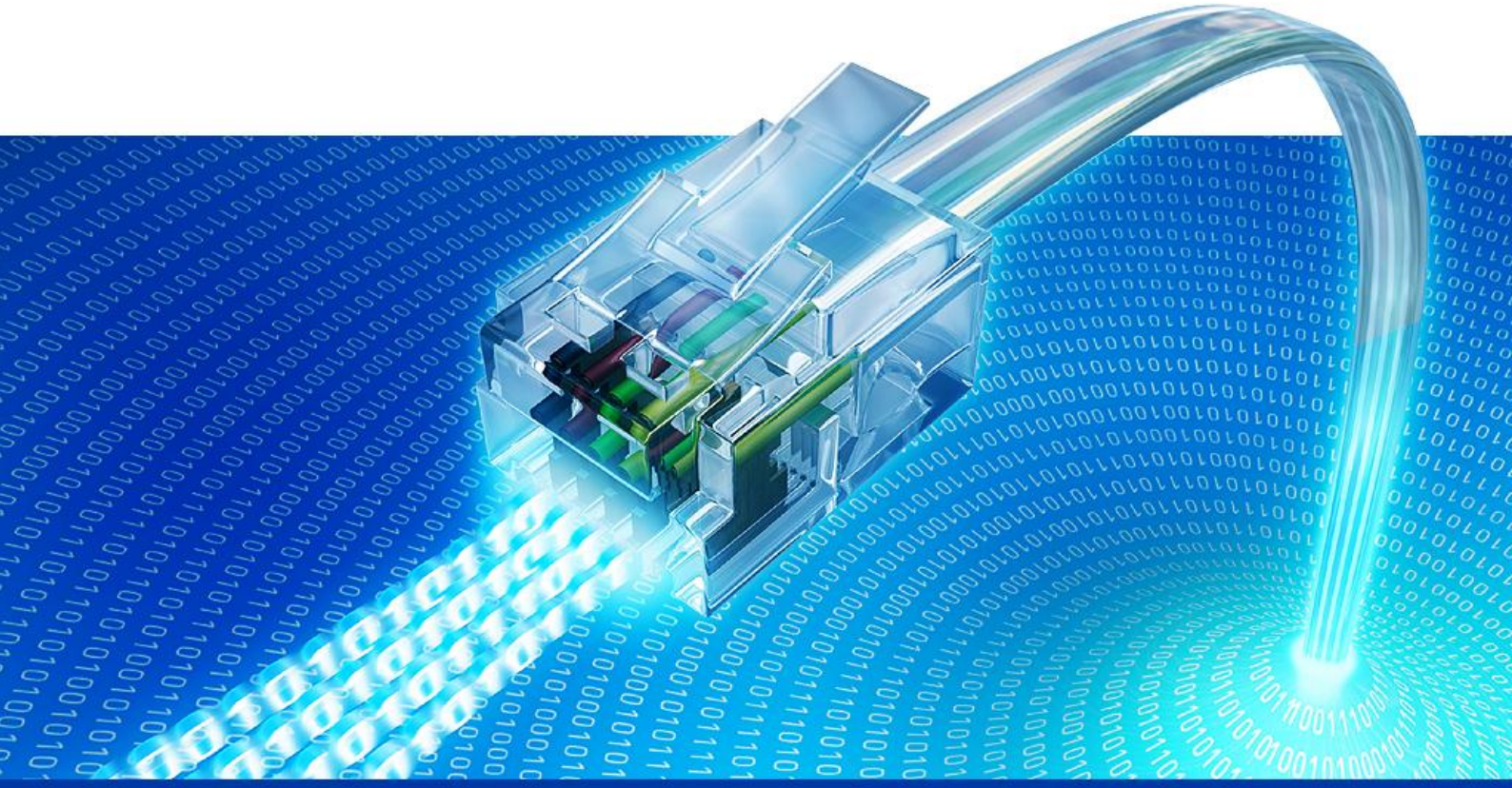
Technische Hochschule Rapperswil

1. Oliver Zürcher

Wie bei der Semesterarbeit, war mein primäres Ziel eine Arbeit in .NET zu schreiben, um wertvolle Erfahrungen für das bevorstehende Berufsleben zu sammeln.
Herr Stettler bat uns einen Slot an, um die Bachelorarbeit an seinem Institut zu absolvieren. Die Idee war, die vorangegangene Semesterarbeit „Captive Portal Load Generator" weiter zu führen. Ich freute mich sehr über das Angebot. Nach einem kurzen Gespräch mit meinem Partner entschieden wir uns schnell, diese Aufgabe anzunehmen.
Die Arbeit ermöglichte es mir, neue Erfahrungen im .NET Bereich zu sammeln. Für die Persistierung konnte das neue Entity Framework 4 eingesetzt werden, welches viele wichtige Neuerungen mit sich bringt. Durch den Einsatz des neuen EF4 lief ein Teil der Software unter der neuen .NET 4.0 Runtime, welche noch nicht von Mono unterstützt wird. Es war interessant mit 2 verschiedenen Runtimes zu arbeiten, welche miteinander kommunizieren müssen.
Bei der Webapplikation entschieden wir uns, ein Javascript Framework einzusetzen, um den Benutzer ein interaktives Userinterface anzubieten. Wir waren uns beide bewusst, dass dieser Entscheid einiges an Einarbeitungszeit mit sich bringen wird. Javascript existiert schon relativ lange und trotzdem erfährt es heutzutage immer wieder Aufschwünge.
Wir konnten zwar wertvolle Erfahrungen mit Javascript sammeln, jedoch erwies sich die Entscheidung mit Javascript zu arbeiten in meinen Augen als schlecht. Der Einzige Vorteil ist der etwas „dünnere" Client durch den Einsatz von Javascript anstelle von zum Beispiel Silverlight. Ansonsten lebten wir nur mit den Nachteilen der Scripting-Sprache: Keine Typsicherheit, schwer zu testen, keine wirklich gute IDE vorhanden etc. Zu guter letzt erwies sich das ExtJS Framework teils als ziemliches Gebastel aus, was mehrere Forumeinträge von offiziellen Supportern/Entwickler des Frameworks nur bestätigen.
Schlussendlich schaue ich auf eine sehr interessante und lehrreiche Zeit der Bachelorarbeit zurück.

2. Ymyr Osman

Bei der Vergabe der Bachelorarbeiten durch das AVT der HSR wurde uns keine der gwünschten Arbeiten zugeteilt. Darum mussten wir uns nochmals für andere Arbeiten bewerben. Herr Stettler hat gesehen, dass uns keine Arbeit zugeteilt wurde und bot uns einen Slot bei ihm an. Die Idee war die Semesterarbeit Weterzuführen. Nach den Besprechungen mit Michael Schneider klang die Idee sehr interessant und wir entschieden uns dazu diese Idee durchzuführen.
Die Bachelorarbeit durfte ich wieder mit Oliver Zürcher dürchführen wie zuvor die Semesterarbeit. Die Teamarbeit verlief reibungslos in der Semesterarbeit, wie auch jetzt in der Bachelor arbeit. Die Aufgaben, Abstimmung, Aufteilung , Kommunikation, etc. funktionierte und ich würde wieder eine Arbeit mit Oliver schreiben.
Die Arbeit selber ermöglichte es mir neue Erfahrungen im .NET bereich und in Internettechnologien zu sammeln.
ASP.NET wollte ich seit langem ein Übungsprojekt machen um dies kennen zu lernen, jedoch verschob ich es immer wieder nach hinten. Mit der Bachelorarbeit kam dies sehr gelegen.
Im bereich der Internettechnologien wurde mit dem ExtJs Framework gearbeitet. Bisher brauchte ich Javascript selber ohne Framewokr oder anderen Libraries. Bei m ExtJs Framework habe ich mit der Arbeit festgestellt das für kleine einfache sachen doch mehr Zeit beansprucht wird weil das Framework im weg steht oder das die Dokumentation nicht ganz vollständig ist.
Das interessante im Projekt mit ASP.NET und Javascript war, dass die Requests an den Controller alle über AJAX gemacht werden. ASP.NET ist nicht nur für eine Javascript oberfläche Konzpiert worden, jedoch zeigte das MVC 2.0 von ASP.NET das die View gut ausgewechselt werden konnte.
Das neu gelernte ist für mich wertvoll den ich kann es für die Zukufunt mitnehmen.
Im ganzen bin ich sehr zufrieden mit dem Projekt. Wir haben vieles erreicht und ich denke das Ergebnis kann in zukunft produktiv eingesetzt werden.



5. Vorarbeit

Network Testcase Engine

Bachelorarbeit FS 2010

Technische Hochschule Rapperswil

1. Einleitung

Vor dieser Bachelorarbeit wurde im Rahmen einer Semesterarbeit ein Prototyp, der Captive Portal Load Generator, entwickelt. Für das Verständnis dieser Arbeit ist eine kurze Zusammenfassung dieser Arbeit notwendig.

2. Ausgangslage

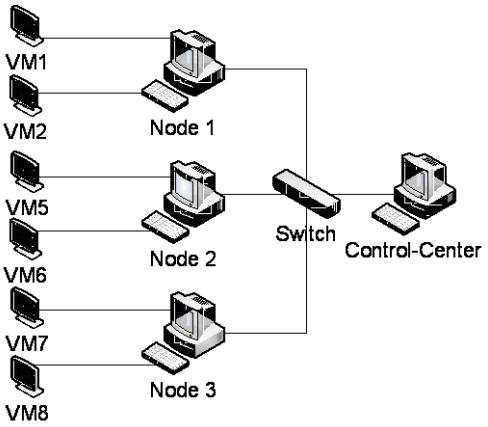
Das INS hat 2003 für den Flughafen Zürich die Internet Access Plattform MPP entwickelt. Seither hat sich der Kundenstamm auf Hochschulen, Grossfirmen und Service-Provider vergrössert. Auch die Anzahl Clients ist an diesen Hotspots in den letzten Jahren regelrecht explodiert, was zu einem sehr starken Anstieg der Last auf den Portalen geführt hat.
Durch den grossen Zuwachs verkompliziert sich die Durchführung von Tests gegen das MPP. Die Anzahl Clients für ein in der Realität entsprechendes Szenario ist bereits zu gross, um solche Tests von Hand effizient durchführen zu können.
Die Aufgabe besteht nun darin, eine Software zu entwickeln, welche es ermöglicht, tausende Clients zu simulieren, um realitätsnahe Szenarien für Tests zu ermöglichen.

3. Vorgehen / Technologien

Es existieren viele verschiedene Produkte zur Virtualisierung auf dem Markt. Es wurden 5 bekannte Lösungen ausgewählt (VMWare, QEMU, Xen, KVM und UML), um sie in einer detaillierten Analyse genauer zu betrachten. Bei möglichst wenig Ressourcenbelastung möglichst viel Leistung aufweisen war eines der wichtigsten Auswahlkriterien.
In einer weiteren Analyse wurden die verschiedenen Charaktere, welche sich am MPP einloggen und Last generieren analysiert und definiert.
Nach diesen beiden Analysen wurde die Software Architektur festgelegt. Es handelt sich um ein verteiltes System, welches von einem Punkt aus gesteuert werden kann.
Als Programmiersprache wurde C# mit dem .NET Framework von Microsoft gewählt. In Kombination mit dem Mono-Projekt lassen sich .NET Assemblies unter Linux ausführen.

4. Ergebnisse

Das verteilte System besteht aus verschiedenen Komponenten, welche von einem Punkt aus gesteuert werden (Control-Center):



4.1 VMRemoteControl

Die Konsolenanwendung VMRemoteControl läuft auf jedem Computer (Node), welcher seine Hardware für die Testszenarien zur Verfügung stellt. Die Applikation bietet Remote-Services zur Verwaltung und Steuerung von virtuellen Maschinen auf dem System. Für jede VM wird das vorkonfigurierte Basisimage verwendet (Debian Linux). Für die Nodes wird Ubuntu als Betriebssystem verwendet.

4.2 LoadGenerator

Die LoadGenerator Anwendung ist ebenfalls eine Konsolenanwendung. Sie ist im Basisimage für die virtuellen Maschinen vorinstalliert und wird beim Systemstart automatisch gestartet. Das Control-Center weist dem LoadGenerator einen Surfertypen zu, welcher dynamisch zur Laufzeit nachgeladen wird. Dieser generiert nach dem Startsignal vom Control-Center die Last gegen das MPP.

4.3 Control-Center

Das Control-Center ist eine GUI Applikation, welche dem Benutzer eine Eingabemaske zur Verwaltung der verschiedenen Szenarien zur Verfügung stellt.
Zur Findung aller Nodes im Netzwerk, welche für die Virtualisierung der Clients zuständig sind, werden per Multicast alle Nodes getriggert, sich beim Control-Center anzumelden. Diese teilen bei der Registrierung ihre verfügbaren Ressourcen dem Control-Center mit. Wenn genügend Ressourcen für das konfigurierte Szenario zur Verfügung stehen, kann das Szenario im Control-Center gestartet werden. Es kann wahlweise eine Anzahl an Durchläufen konfiguriert werden, oder fortlaufend, bis der Benutzer das Szenario stoppt.

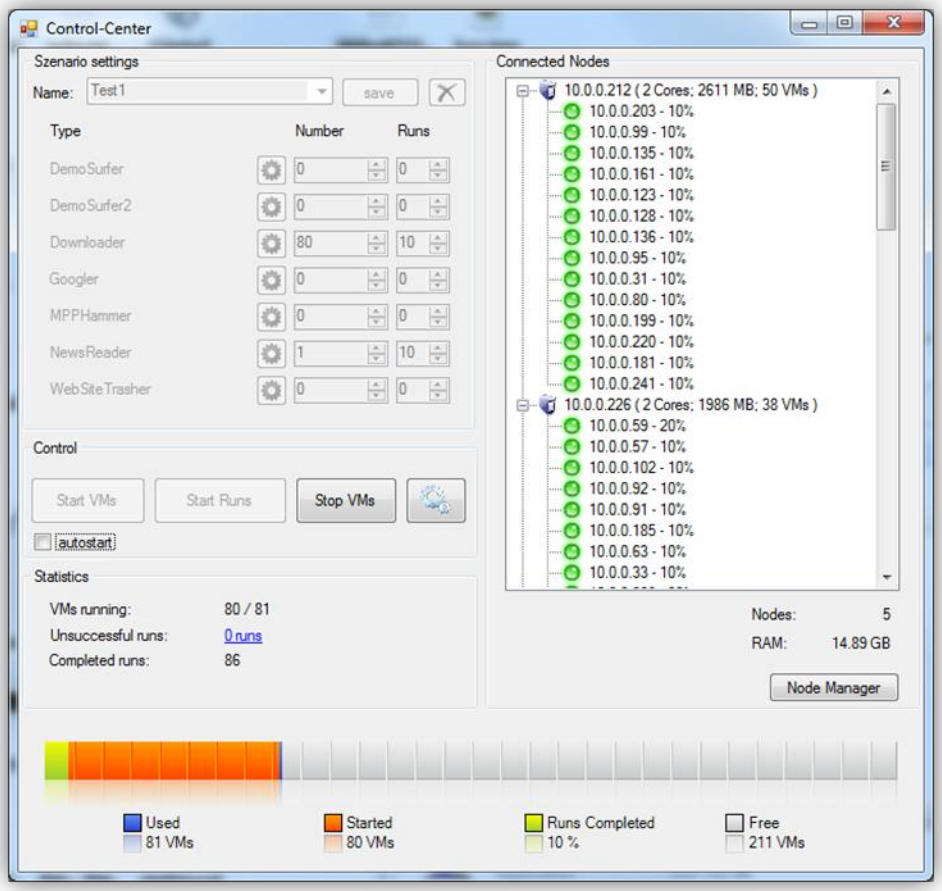
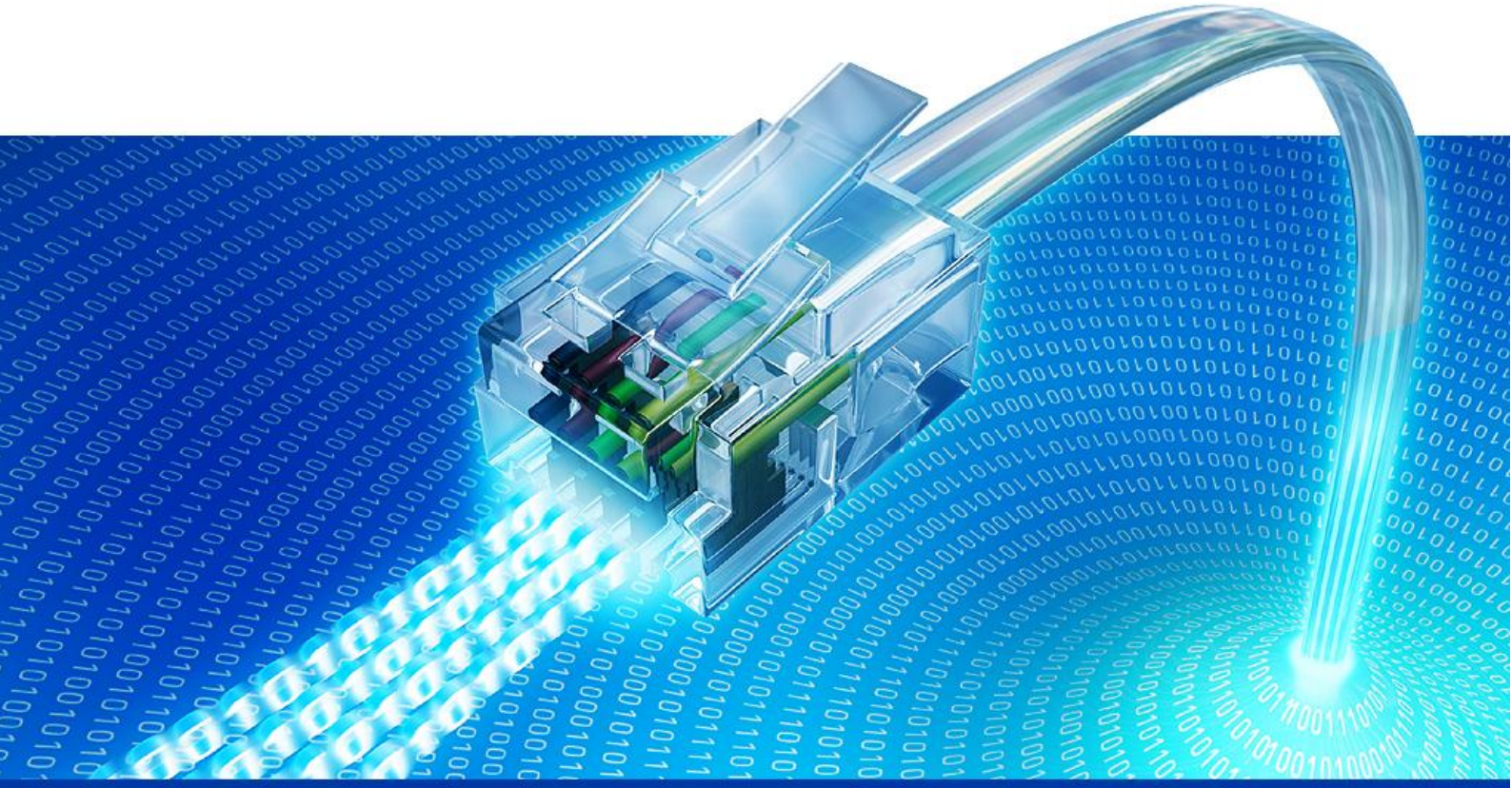


Abbildung 11: Control-Center GUI

5. Ergebnis der Semesterarbeit

Die entwickelte Software ist ein lauffähiger Prototyp. Durch die Load Generator API können weitere Surfertypen entwickelt werden. Die Architektur ist so aufgebaut, dass diese dynamisch geladen werden. Bestehender Code muss beim Hinzufügen weder geändert noch neukompiliert werden. Das Captive Portal Load Generator Projekt bietet eine hervorragend skalierende Lösung zum effizienten Testen des MPPs.



6. Projektplan

Network Testcase Engine

Bachelorarbeit FS 2010

Technische Hochschule Rapperswil

1. Projektorganisation

Das Team besteht aus 2 einander gleich gestellten Teammitgliedern. Herr Prof. Beat Stettler und Herr Michael Schneider fungieren als Kontrollorgan und unterstützen das Projekt bei offenen Fragen. Das Team hat sich folgende Zeiten für das Projekt reserviert:

Dienstag:	08:10 – 16:50
Mittwoch:	08:10 – 16:10
Donnerstag:	12:50 – 15:50
Freitag:	12:50 – 15:50

2. Organisationsstruktur

Alle Teammitglieder erhalten mehrere Aufgabenbereiche zugeteilt. Sie übernehmen dabei die Führungsrolle in diesem Bereich. Wenn Termine nicht eingehalten werden können, oder sonstige Probleme auftreten, sind sie dafür verantwortlich, dies an den Sitzungen zu thematisieren. Die einzelnen Aufgaben innerhalb eines Bereiches werden unter allen Mitgliedern aufgeteilt.

Name	Tätigkeit	Verantwortung
Ymyr Osman	Entwicklung	Projektmanagement, Virtualisierung, Webinterface
Oliver Zürcher	Entwicklung	Dokumentation, Analyse, Architektur, Konfigurationsmanagement

2.1 Externe Schnittstellen

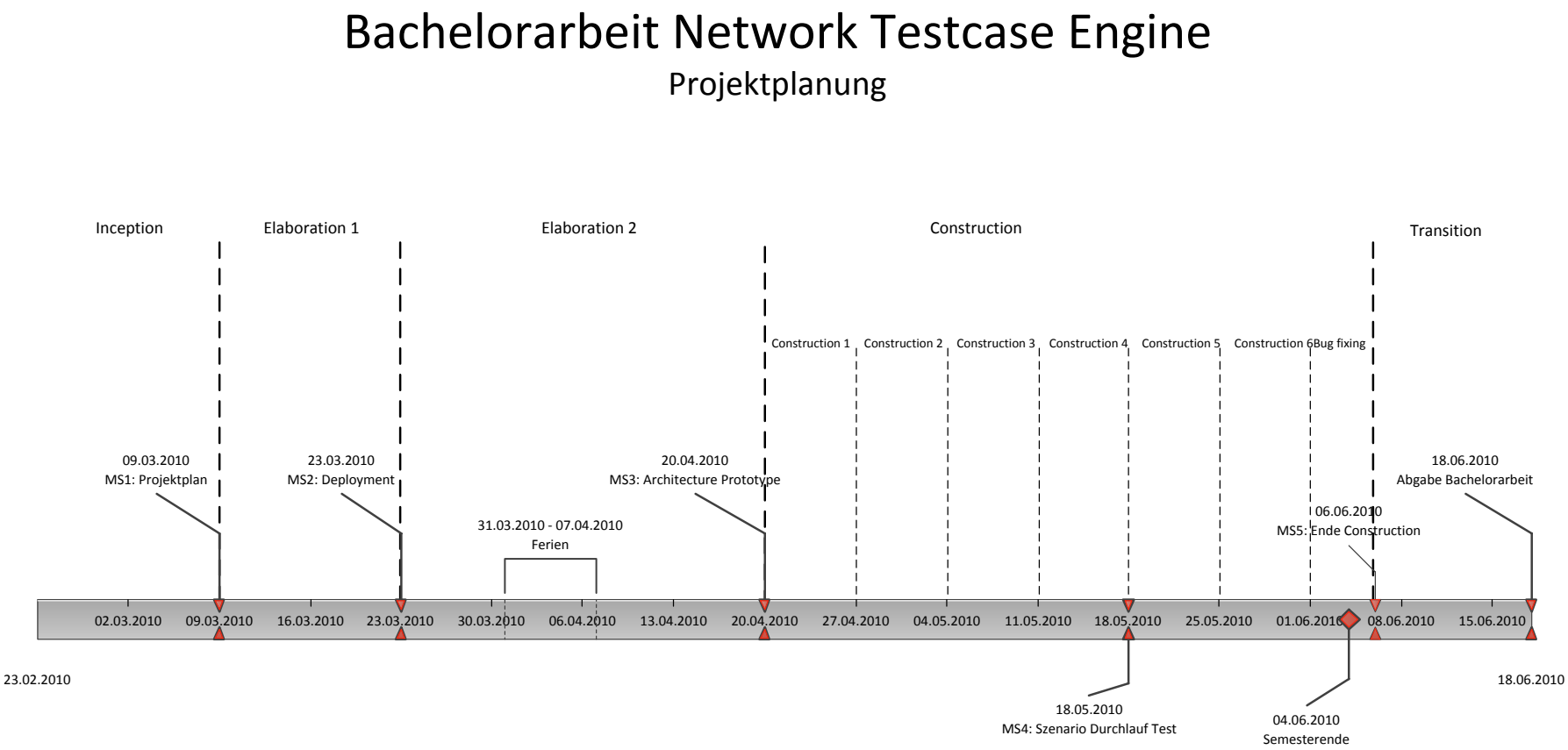
Name	Tätigkeit
Beat Stettler	Betreuer
Michael Schneider	Betreuer
Andreas Steffen	Gegenleser
Markus Vögtlin	Experte

3. Management Abläufe

3.1 Projekt Kostenvoranschlag

Dem Projekt stehen pro Woche und pro Teammietglied ca. 20 Stunden zur Verfügung für die ersten 14 Wochen. Die letzten zwei Wochen wird täglich zu 8h gearbeitet. Das heisst je Teammitglied stehen pro Woche 40h zur Verfügung. Das Arbeitspensum pro Mitglied kann bei Problemen um 2-3 Stunden erhöht werden.
Der Projektstart wurde auf den 23. Februar 2010 und der Abgabe-Termin wurde auf den 18. Juni 2010 festgelegt.

3.2 Zeitplan


Bachelorarbeit Network Testcase Engine
Projektplanung
Inception
Elaboration 1
Elaboration 2
Construction
Transition
Construction 1
Construction 2
Construction 3
Construction 4
Construction 5
Construction 6
Bug fixing
09.03.2010
MS1: Projektplan
23.03.2010
MS2: Deployment
20.04.2010
MS3: Architecture Prototype
31.03.2010 - 07.04.2010
Ferien
06.06.2010
MS5: Ende Construction
18.06.2010
Abgabe Bachelorarbeit
02.03.2010
09.03.2010
16.03.2010
23.03.2010
30.03.2010
06.04.2010
13.04.2010
20.04.2010
27.04.2010
04.05.2010
11.05.2010
18.05.2010
25.05.2010
01.06.2010
08.06.2010
15.06.2010
23.02.2010
18.06.2010
18.05.2010
MS4: Szenario Durchlauf Test
04.06.2010
Semesterende

3.2.1 Construction Phase

In der Construction Phase wird SCRUM ähnlich mit einem Product Backlog, welches alle Features enthält, gearbeitet. Am Anfang jeder Construction Phase werden Features aus dem Product Backlog ausgewählt, welche in der folgenden Iteration erledigt werden.
Für die Qualitätssicherung wird am Ende jeder Construction Phase ein kurzes Review durchgeführt, welches schriftlich festgehalten wird.

3.2.2 Meilensteine

Woche	Inhalt	Meilenstein
3	Anforderungen definiert, Projektplan, Analyse Domänenmodel	MS1: Anforderungen definiert
5	Deployment auf allen Tiers abgeschlossen	MS2: Deployment
8	Architekturprototyp	MS3: Architecture Prototype
12	Prototyp, aus dem Webinterface kann ein Szenario gestartet werden.	MS4: Szenario Durchlauf Test
15	Alle Features fertig implementiert, Unit Testing	MS5: Ende Construction
16	Schlussberichte, Abstract, Management Summary, Glossar, Literaturverzeichnis	MS 6: Abgabe Bachelorarbeit

3.3 Besprechungen

Das Team trifft sich mit den Betreuern jeweils dienstags um 13:30 Uhr.
Besprechungen werden alle Protokolliert und sind im Anhang zu finden.

3.4 Releases

Release	Woche	Beschreibung
Beta	8	Architekturprototyp
Final	15	Alle Features implementiert

3.5 Artefakte

Meilenstein	Artefakte
MS1	Projektplan Analyse
MS2	Deployment Skripte Software Architectural Document
MS3	Architektur Prototyp über alle Tiers

	Software Architectural Document
MS4	Bachelorarbeit.pdf

4. Risiko Management

Nr	Titel	Beschreibung	max. Schaden [h]	Eintrittswahrscheinlichkeit	gewichteter Schaden [h]	Massnahmen zur Vermeidung / Verhinderung	Vorgehen bei Eintreffen
R1	Ausfall Arbeitsstation	HW eines Projektmitglieds fällt aus	4	5%	0.2	- Aktuelle Hardware verwenden (Aktueller Zustand der Hardware überprüfen und Mängel, sofern möglich, beheben) - Aktuelle Software verwenden (Updates, Virenschutz)	- Arbeitsplatz Rechner der HSR verwenden - Eigener PC zuhause verwenden
R2	Ausfall Netzwerkinfrastruktur	Netzwerk an der HSR oder Anbindung des MPPs fällt aus	48	5%	2.4	- Datentransfermöglichkeiten innerhalb des Teams bereithalten: USB, CD-Rom	- Daten über anderes Netzwerk oder Medien austauschen
R3	Datenverlust	Datenverlust durch HW oder SW Fehler	720	1%	7.2	- Lokale Kopie der Daten - Daily Backup auf externe redundante Storage - SVN / TFS	-Daten aus dem Backup wiederherstellen - Wenn Daten verloren sind, abklären ob das Projekt gefährdet ist
R4	Ausfall eines Projektmitglieds	Ein Projektmitglied erleidet einen Unfall oder wird krank	360	2.5%	9	-Verantwortlichkeit im Team definieren	- Arbeit innerhalb des Teams aufteilen
R5	Streit im Projektteam	Es entstehen Streitigkeiten im	24	5%	1.5	- Konstruktive Kritik und Kommunikation im Projektteam	- Uhrsachen früh Erkennen und

		Projektteam				pflegen und fördern	vermeiden
R6	Fehleinschätzungen des Aufwandes	Zeitplan wird nicht eingehalten weil der Aufwand falsch eingeschätzt wurde	30	10%	3	-Kritische Arbeitspakete frühzeitig bearbeiten -Wöchentliche Sitzung mit Betreuern	- Aussortieren gewisser Funktionen und Features in den Sitzungen
R7	Fehleinschätzung durch Annahmen	Falsche Implementationen durch Annahmen	15	5%	0.75	-Sobald eine Annahme getroffen wird mit Teammitglied absprechen / nachfragen	- Teamdiskussion einberufen
R8	Probleme mit Technologien	Einarbeitung in ASP.NET	30	10%	3	- Dokumentation und Anleitungen im Internet und Sachbüchern suchen	- Mehr Zeit in Einarbeitung investieren. Projektplan muss eingehalten werden
R9	Probleme bei der Kommunikation	Befehle können nicht gesendet oder Empfangen werden	30	10%	3	- Verschiedene Mechanismen für eine Kommunikation in Betracht ziehen und studieren	- Kommunikationslösung austauschen durch eine andere Möglichkeit
Sum			**1261**	**56%**	**30.05**		

Tabelle 1: Risikoanalyse

Unsere Zeitplanung enthält eine Reserve von ca. 36 Stunden, was etwa 18 Stunden pro Teammitglied entspricht. Die Risiko Analyse zeigt, dass wir ein Risiko von 30.05 Stunden haben. Unsere Zeitplanung enthält daher eine Reserve von 36 Stunden. Durch diese Reserve können wir den gewichteten Schaden abdecken.

5. Infrastruktur

5.1 Räumlichkeiten

Besprechungen finden im Büro 6.001 der HSR statt. Es wird im Bachelorarbeitszimmer 1.262 gearbeitet oder von Zuhause aus. Je nach Arbeitssituation.

5.2 Hardware

Primär arbeitet jeder mit seinem persönlichen Notebook. Bei Ausfall sind die HSR PCs vorhanden. Als Versionsverwaltung dient uns ein Team Foundation 2010 Server.

5.3 Software

Betriebssysteme	- Microsoft Windows 7 - Debian 5.0 „Lenny“ - Ubuntu 10.4 „Lucid Lynx“
Programmiersprache	- Microsoft C#.NET / Mono
Entwicklungsumgebung	- Microsoft Visual Studio Team System 2010 - Monodevelop 2.0
Versionsverwaltung	- Microsoft Team Foundation Server 2010 RC
Datenbank	- Microsoft SQL 2008
Dokumentation	- Microsoft Office 2010

5.4 Backup

Der Team Foundation Server läuft auf einem Microsoft Windows 2008 Enterprise Server, welcher in Frankfurt am Main im link11 Rechenzentrum steht. Auf diesem Server werden zwei Mal täglich „shadow copies“ für kleinere Probleme erstellt. Der SQL und Sharepoint Server werden jeweils in der Nacht von Sonntag auf Montag voll und in jeder Nacht inkrementell gesichert. Die Backups werden für 2 Wochen auf dem Server selbst gespeichert und in jeder Nacht auf ein Rechner in der HSR kopiert.

5.5 Kommunikation

Der Team Foundation Server unterstützt uns bei der Kommunikation. Es existiert eine Team Seite (Sharepoint Portal) in der alle Dokumente verwaltet werden können. Es werden alle Artefakte „versioniert“, nicht nur der Sourcecode.
Zusätzlich unterstützen uns folgende Kommunikationsmittel für einen erfolgreichen Projektverlauf:

- Besprechungen
- E-Mail
- MSN-Messenger

- Teamspeak (VOIP)

6. Qualitätsmassnahmen

6.1 Dokumentation

Grosser Wert wird darauf gelegt, dass die Dokumentation stets aktuell und vollständig ist. Änderungen müssen möglichst schnell von den Teammitgliedern eingecheckt werden. Falls es zu Konflikten kommt, welche nicht ohne weiteres zusammengeführt werden können, muss das Dokument mit einem Suffix im Format<DOKUMMMENT_NAME>_<*NAME*> auf der Team Seite genannt werden. Diese Konflikte werden dann im Team besprochen und behoben.

6.2 Besprechungsprotokolle

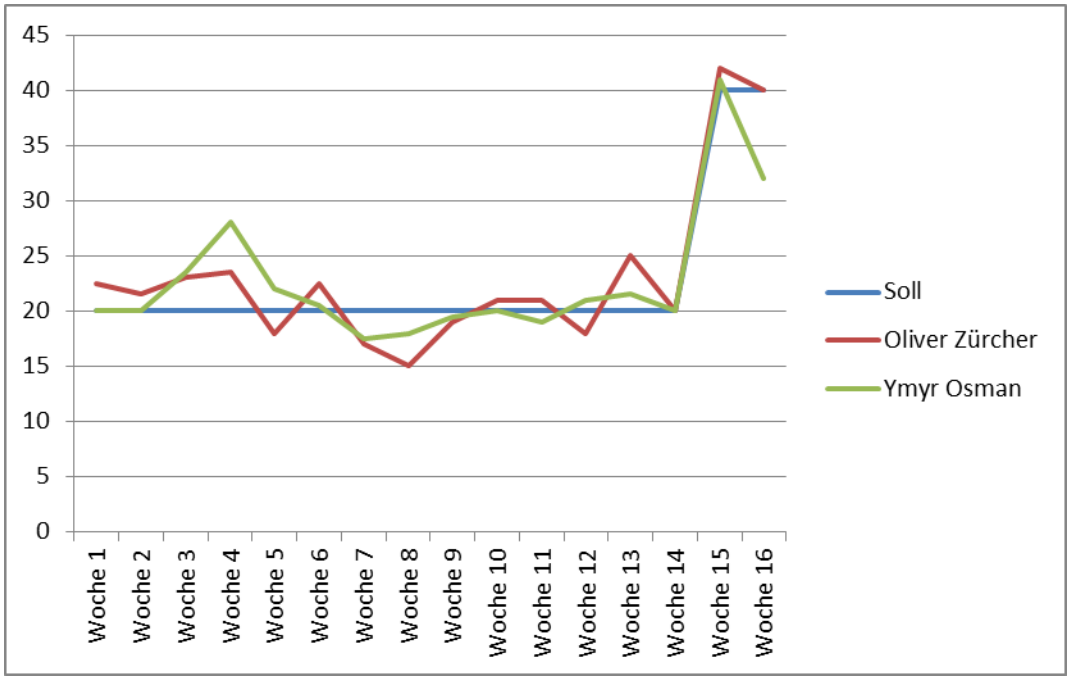
Sitzungen werden jeweils von einem Teammitglied protokolliert. Mit dieser Massnahme möchten wir besprochenes schriftlich festhalten (Kritik, Vorschläge, Anregungen usw.).

6.3 Versionskontrolle

Durch den Einsatz eines Team Foundation Servers, ist die Versionskontrolle direkt in den von uns eingesetzten Programmen integriert (Word, Excel, Visual Studio, Explorer usw.). Alle Artefakte werden während dem gesamten Projektverlauf versioniert.

7. Projektauswertung

7.1 Zeitauswertung



7.2 Codeauswertung

Um den Fortschritt und die Qualität der Entwicklung zu kontrollieren, wird bei jedem Review eine Codeauswertung durchgeführt. Dabei sind verschiedene Indikatoren, welche mit Visual Studio 2010 Team System generiert werden können, Anhaltspunkte für die Codequalität. Die Anzahl an Codezeilen spielt dabei eine relativ unwichtige Rolle.
"Measuring programming progress by lines of code is like measuring aircraft building progress by weight." Bill Gates, Microsoft
Die Codeauswertung umfasst folgende Punkte:

- Maintainability Index

  Indikator von 0 bis 100 welcher die Wartbarkeit der enthaltenen Typen misst. Dabei gilt 0 als schlecht Wart bar und 100 als hoch Wart bar
- Cyclomatic Complexity

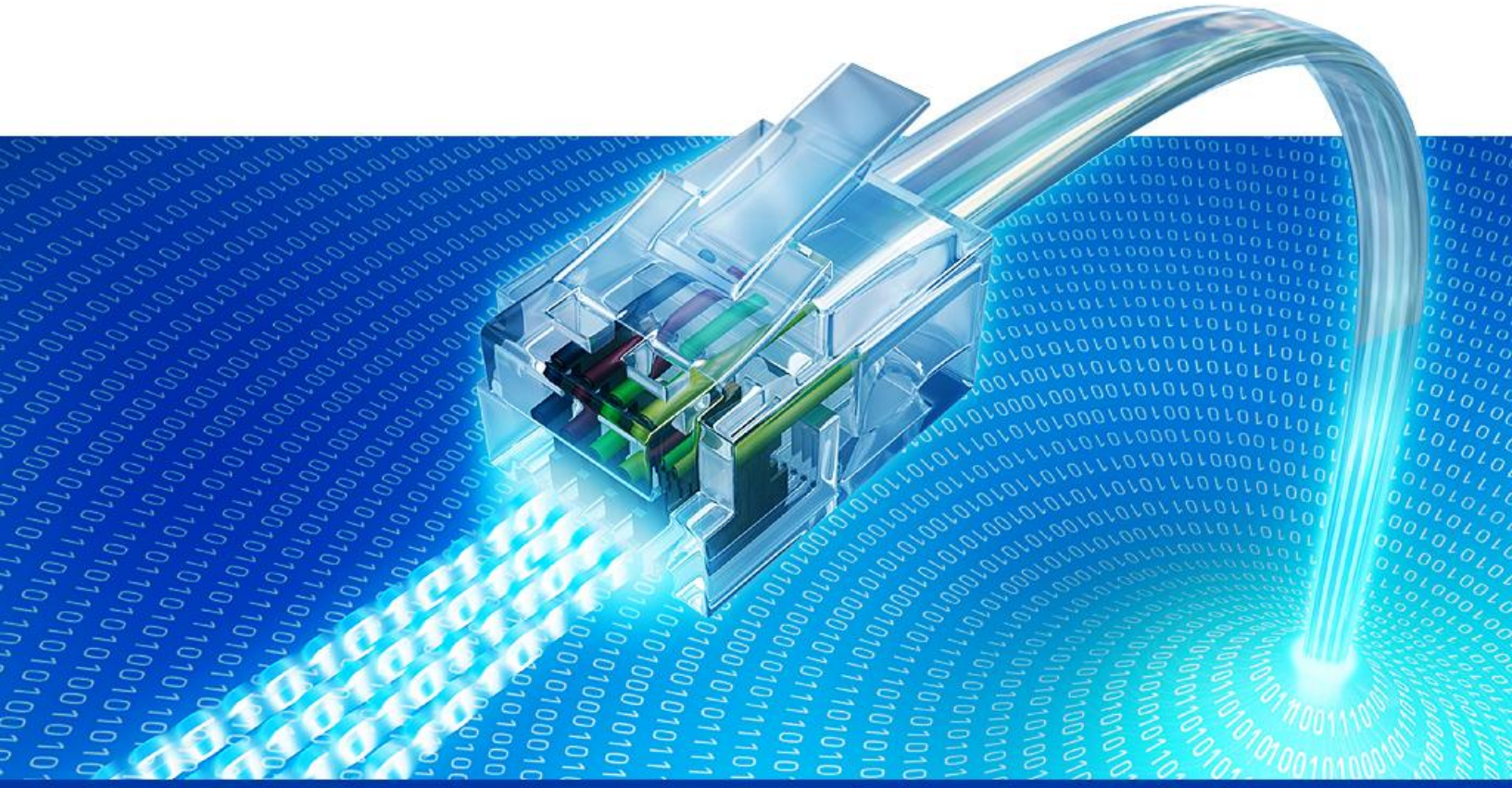
  Dieser Wert misst auf jeder Ebene die gesamte Anzahl der individuellen Wege im Code. Dieser berechnet sich durch Entscheidungspunkte. Das sind Anweisungen wie z.B. if-Blöcke, switch-Anweisungen oder Schleifen. Hier gilt ein Tieferer Wert als besser.
- Depth of Inheritance

  Dieser Wert zeigt zeigt, wie viele Typen sich über dem aktuellem Typen in der Vererbungshierarchie befinden. Eine hohe Vererbung kann ein Anzeichen von over-engineering sein. Das heisst, die Komplexität des Testens oder die Wartbarkeit der Applikation steigt.

- Class Coupling

  Dieser Wert zeigt die gesamte Anzahl Abhängigkeiten eines Typs. Dabei wird geachtet das die primitiven Datentypen wie z.B. Integer oder String nicht mitgezählt werden. Ein tiefer Wert heisst hierbei, dass dieser Typ wiederverwendet werden könnte.

Am Ende jeder Construction Phase werden die Werte überprüft. Wenn die Werte nicht im grünen Bereich sind, muss dies im Bericht notiert werden, damit dies in der nächsten Construction Phase verbessert werden kann.



7. Analyse

Network Testcase Engine

Bachelorarbeit FS 2010

Technische Hochschule Rapperswil

1. Domainanalyse

1.1 Strukturdiagramm

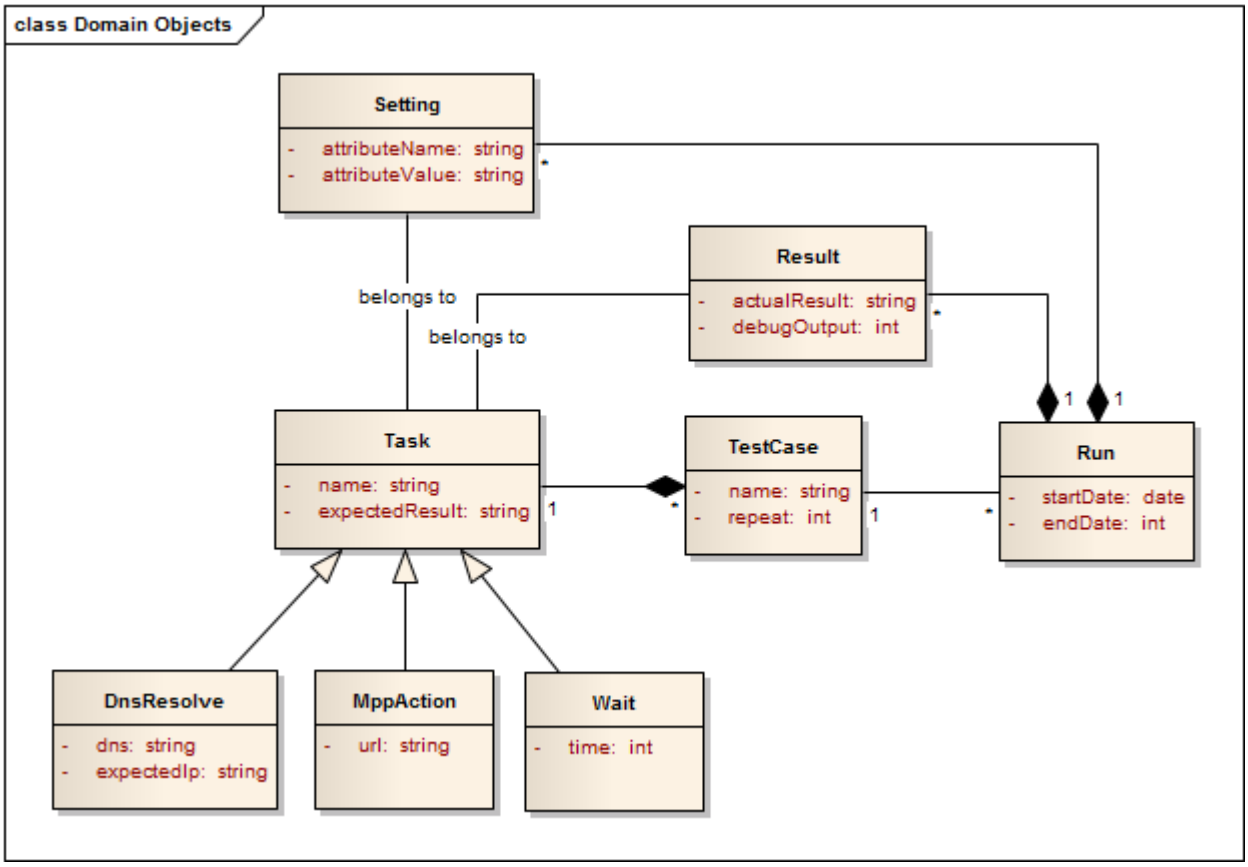


Abbildung 12: Domainanalyse Strukturdiagram

1.2 Konzeptbeschreibung

1.2.1 Testcase

1	TestCase	
Beschreibung	Ein TestCase beinhaltet die Informationen, wie ein Ablauf erfolgt.	
Attribute	Name	Name des TestCases
	Repeat	Wie oft der Ablauf wiederholt wird
Beziehungen	Ein TestCase generiert mehrere Berichte Ein TestCase enthält mehrere Tasks	

1.2.2 Task

2	Task	
Beschreibung	Ein Task ist ein Element im TestCase	
Attribute	Name	Name des Tasks
Beziehungen	Abstrakte Basisklasse von „DNSResolve“, „MPPAction“, „Wait“ Ist beliebig oft in einem TestCase enthalten	

	Ein Task enthält beliebig viele Attribute Setting von RuntimeConfiguration gehören zu einem Task

1.2.3 DNSResolve

3	DNSResolve	
Beschreibung	Spezialisierung von Task. Überprüft ob eine Name Resolution möglich ist	
Attribute	DNS	URL welche aufgelöst werden soll
	ExpectedIP	IP Adresse, welche als Ergebnis erwartet wird
Beziehungen	Abgeleitet von Task	

1.2.4 MPPAction

4	MPPAction	
Beschreibung	Spezialisierung von Task. Loggt sich beim MPP ein oder aus	
Attribute	URL	Die URL für die Landingpage des MPP's
Beziehungen	Abgeleitet von Task	

1.2.5 Wait

5	Wait	
Beschreibung	Spezialisierung von Task. Wartet für eine bestimmte Zeitspanne	
Attribute	Time	Wartezeit in Sekunden
Beziehungen	Abgeleitet von Task	

1.2.6 Bericht

6	Bericht	
Beschreibung	Enhält die Meldungen, Fehelr oder Erfolg, eines TestCases	
Attribute	EndDate	Startzeitpunkt der Ausführung vom TestCase
	StartDate	Endzeitpunkt der Ausführung vom TestCase
Beziehungen	Ein TestCase beinhaltet mehrere Reports Enthält eine RuntimeConfiguration	

1.2.7 RuntimeConfiguration

7	RuntimeConfiguration	
Beschreibung	Enhält die Laufzeitdaten des TestCases inklusive den Setting je Task	
Attribute		
Beziehungen	Gehört zu einem Bericht	

1.2.8 Setting

8	Setting	
Beschreibung	Enhält die einstellungen von einem Task	
Attribute	Name	Name der Einstellung des Tasks
	Value	Zugewiesener Wert
Beziehungen	Ein Setting exisitert einmal pro Task	

2. Technologieanalyse

2.1 Entity Framework

Das Entity Framework ist ein ORM (Object Relational Mapping) Framework von Microsoft. Mithilfe des EF können Datenbanktabellen an Objekte gebunden werden. Das EF befindet sich im RC Stadium und wird mit .NET 4.0 und Visual Studio 2010 im April released. Folgende Vorteile bietet der Einsatz des Entity Frameworks:

- Erleichterung der Arbeit mit der Datenbank im allgemeinen
- Zugriff auf Datenbank über Objekte, keine SQL Kenntnisse notwendig
- Datenhaltung ist an einer Stelle vorhanden. Damit wird dem „DRY“ Prinzip gefolgt
- Methoden zur Datenmanimpulation werden automatisch generiert
- Datenbank kann nun generiert werden
- Neue Code-First Ansätze, um technologieneutrale Klassen mit dem EF zu persistieren

Die folgenden Abschnitte zeigen Beispiele wie mit dem Entity Framework gearbeitet wird.

2.1.1 Model-First

Bisher konnte man nur aus einer existierenden Datenbank ein Model erstellen. Dieses wurde in Visual Studio bearbeitet und angepasst um es im Programm gebrauchen zu können.
Nun besteht die Möglichkeit, die Datenbank selber zuerst in Visual Studio zu designen. Wenn das Design fertig ist, kann das Model daraus generiert werden.

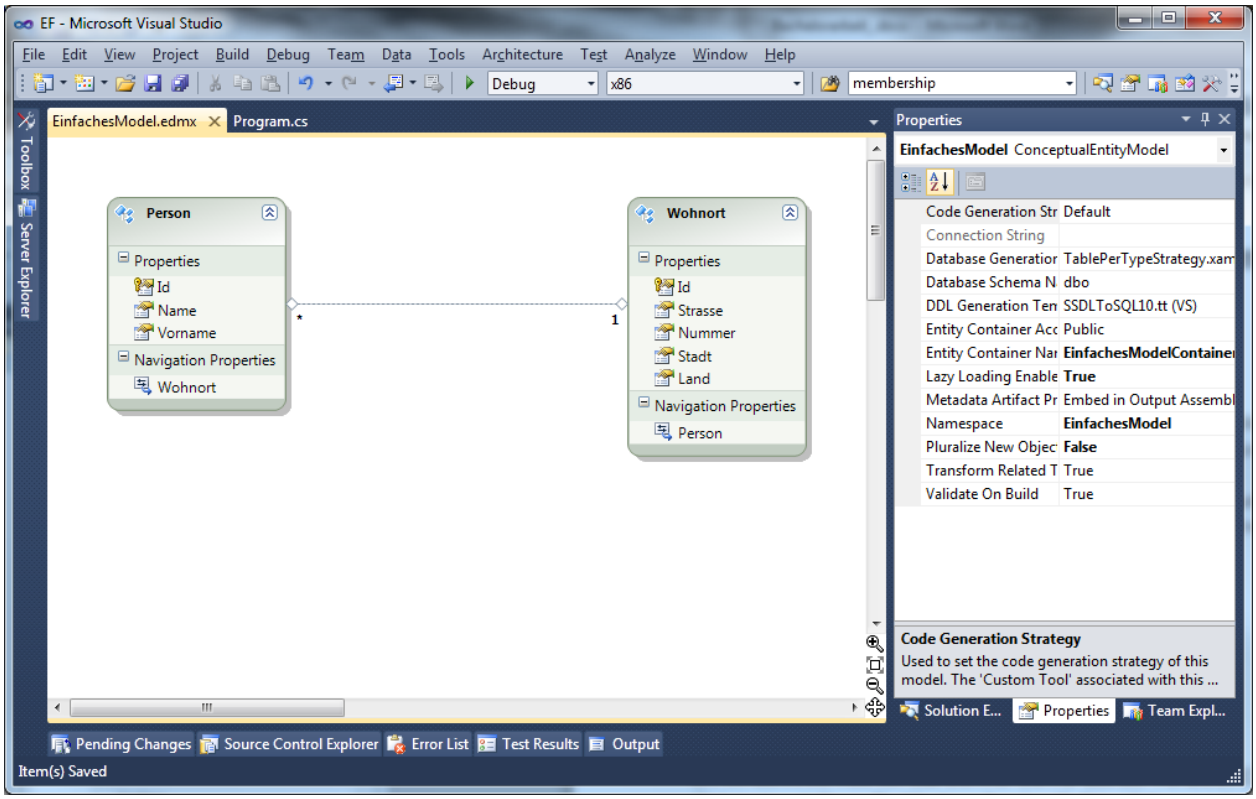


Abbildung 13: Der neue Visual Studio Designer

2.1.1.1 Codebeispiel

Dieses Beispiel zeigt die Erzeugnung von Zwei Datenbankeinträgen. Ein neuer Eintrag für die Person Tabelle und einer für die Wohnort Tabelle gemäss dem obigen Konzeptionellen Design.

```
class Program
{
    static void Main(string[] args)
    {
        using (var container = new EinfachesModelContainer())
        {
            Person p = new Person() { Name = "Müller", Vorname = "Hans" };

            Wohnort w = new Wohnort();
            w.Strasse = "Musterstrasse";
            w.Nummer = 25;
            w.Stadt = "Musterstadt";
            w.Land = "Musterland";

            container.PersonSet.AddObject(p);
            container.WohnortSet.AddObject(w);
            container.SaveChanges();
        }
    }
}
```

2.1.2 Code First

Der Codefirst Ansatz bietet dem Entwickler die Möglichkeit, die Klassen, welche später persistiert werden sollen, zuvor technologieneutral zu entwickeln. Es sind keine Technologiespezifische Attributierungen wie in früheren Versionen notwendig. Durch diese klare Trennung, müssen keine

Änderungen an den Business Klassen vorgenommen werden, wenn im Nachhinein entschieden wird, die Daten mit dem EF zu persistieren.
Das Mapping wird beim Code First Ansatz in einer Mapperklasse definiert, welche von der generischen Klasse `EntityConfiguration` ableitet.
Zusätzlich zur Mapperklasse muss eine Kontextklasse implementiert werden, welche alle Objekte verwaltet und die Objektmodelle zur Verfügung stellt. Diese Klasse leitet von der Klasse `ObjectContext` ab.

2.1.2.1 Implementierungsbeispiel Code First

Als erstes müssen die Klassen erstellt werden, welche das Objektmodell repräsentieren.

```
public class Kurs
{
    public Guid Id { get; set; }
    public string Name { get; set; }
    public DateTime Kursausfuehrung { get; set; }
    public ICollection<Durchfuehrung> Durchfuehrungen { get; set; }
}

public class Durchfuehrung
{
    public Guid Id { get; set; }
    public string Titel { get; set; }
    public string Beschreibung { get; set; }
    public string Sprecher { get; set; }
    public Kurs Kurs { get; set; }
}
```

Als nächstes wird der Objektkontext implementiert. Dies ist der Mittelpunkt für den späteren Gebrauch des Entity Frameworks. Hier stellt man die vorhin erstellten Objektmodelle zur Verfügung. Der Objektkontextklasse wird im Konstruktor eine Verbindung zur Datenbank übergeben.
Mittels Properties stellt man die Objektmodelle Durchfuerung und Kurs aus dem Beispiel zur Verfügung:

```
public class KursModel : ObjectContext
{
    public KursModel(EntityConnection connection)
        : base(connection)
    {
    }

    public IObjectSet<Durchfuehrung> Durchfuehrungen
    {
        get
        {
            return base.CreateObjectSet<Durchfuehrung>();
        }
    }

    public IObjectSet<Kurs> Kurse
    {
        get
        {
            return base.CreateObjectSet<Kurs>();
        }
    }
}
```

Der nächste Schritt ist das konkrete Abbilden der Objekte auf das Entity Framework.
Dies geschiet durch das Ableiten der `EntityConfiguration` Klasse. Dabei wird im Konstruktor mittels der zur Verfügung gestellten Methode `Property()` das Mapping sowie auch die Relationen definiert:

```
public class KursMap : EntityConfiguration<Kurs>
{
    public KursMap()
    {
        Property(k => k.Id).IsIdentity();
        Property(k => k.Name).HasMaxLength(50).IsRequired();
        Property(k => k.Kursausfuehrung);

        Relationship(d => d.Durchfuehrungen).FromProperty(k => k.Kurs);
    }
}

public class DurchfuerungMap : EntityConfiguration<Durchfuehrung>
{
    public DurchfuerungMap()
    {
        Property(d => d.Id).IsIdentity();
        Property(d => d.Titel).HasMaxLength(50).IsRequired();
        Property(d => d.Beschreibung);
        Property(d => d.Sprecher).IsRequired();

        Relationship(k => k.Kurs).FromProperty(d => d.Durchfuehrungen);
    }
}
```

Nach diesen Schritten ist alles nötige Implementiert, die Objekte können nun persistiert werden. Dazu muss zuerst ein `ContextBuilder` mit dem eigenen Kontextmodel erstellt werden. Diesem müssen dann die Abbildungen der Objektmodelle mitgeteilt werden, damit er diese kennt und verarbeiten kann.
Damit der ContextBuilder die Datenbank
kennt, wird im Konstruktor eine `SqlConnection` übergeben.

```
static void Main(string[] args)
{
    var builder = new ContextBuilder<KursModel>();
    builder.Configurations.Add(new KursMap());
    builder.Configurations.Add(new DurchfuerungMap());

    var connection = new SqlConnection(@"server=(local);integrated
security=true;database=KursDB");

    using (var context = builder.Create(connection))
    {
        if (context.DatabaseExists())
            context.DeleteDatabase();

        context.CreateDatabase();

        var kurs = new Kurs();
        kurs.Name = "TestKurs";
        kurs.Kursausfuehrung = new DateTime(2010, 10, 10);

        var durchfuehrung = new Durchfuehrung();
        durchfuehrung.Titel = "TestTitel";
```

```
            durchfuehrung.Sprecher = "Hans Muster";
            durchfuehrung.Beschreibung = "Die Beschreibung";
            durchfuehrung.Kurs = kurs;

            kurs.Durchfuehrungen = new List<Durchfuehrung> { durchfuehrung };

            context.Kurse.AddObject(kurs);
            context.SaveChanges();

        }
    }
```

2.1.3 Laden der Daten

Um eine gute Performanz der Applikation zu gewährleisten, können im Entity Framework verschiedene Ladestrategien angewendet werden. Der Unterschied der Strategien ist der Zeitpunkt, an dem die SQL Statements gegen den Server abgesetzt werden. Grundsätzlich existieren zwei Strategien:

Lazy Loading: Beim Lazy Loading werden die Daten, wie der Name es schon vermuten lässt, im letzten möglichen Moment geladen. Wird zum Beispiel im Code oben eine Liste mit allen Kursen geladen, so werden die dazugehörigen Durchführungen nicht abgefragt. Erst beim Zugriff von einem Kurs Objekt auf seine Durchführungen, wird das SQL Statement an den Server geschickt und das Resultat vom Framework auf die entsprechende Klasse abgebildet.
Dies ist dann sinnvoll, wenn nur eventuell und auf wenige Assoziationen einer Entität zugegriffen wird. Die Zeit zum Laden der Daten ist durch die schlankere Abfrage schneller. Wenn aber Daten mit der Lazy Loading Strategie geladen werden, und dann trotzdem auf alle Assoziationen zugegriffen wird, kann dies die Performance verschlechtern.

Eager Loading: Beim Eager Loading werden die Daten auf Vorrat geladen. Ob diese gebraucht werden oder nicht.
In unserem Beispiel würden zu allen Kursen auch ihre Durchführungen geladen werden. Dadurch ist der Zugriff auf die Durchführungen schneller, bei vielen Datensätzen kann aber die Abfrage schnell viel Zeit und RAM in Anspruch nehmen.

Es gibt keine richtige Lösung. Welche Strategie eingesetzt wird, hängt immer vom Anwendungsfall ab. Es ist Sache des Programmierers, sich Gedanken zu machen, was mit den Daten passieren wird und abzuschätzen, welche Strategie sinnvoller ist.

2.1.4 Umsetzung im Projekt

Da in unserem Projekt die Entitätsklassen auch unter Mono und somit der 3.5 Runtime laufen müssen, setzen wir den Code First Ansatz an. Das Mapping wird in der neuen 4.0 Runtime umgesetzt, die Entitätsklassen bleiben Technologieneutral auf der 3.5 Runtime.

2.2 Windows Services

Ein Windows Service soll das Control-Center aus der Vorarbeit Captive Portal Load Generator ablösen. Ein Windows Service läuft im Hintergrund und ist im Konzept mit dem Unix-Daemon sehr ähnlich. Folgende Eigenschaften zeichnen einen Service aus:

- Läuft auch ohne einen angemeldeten Benutzer
- Interagiert in der Regel nicht mit dem Desktop eines Benutzers
- Wird vom Windows Service Manager (services.exe) kontrolliert

Der Sinn eines Services ist, dass ein System bestimmte Dienste bereitstellen kann, ohne dass ein Benutzer diese explizit aufstarten muss. Dienste können beim Windowsstart automatisch gestartet werden, manuell auf Anforderung anderer Programme oder komplett deaktiviert sein. Das Betriebssystem startet Dienste unabhänig davon, ob die graphische Benutzeroberfläche gestartet wurde.

2.2.1 Installation

Dienste können über das MMC-Snap-In „*services.msc*“ verwaltet werden. Über das Kommandozeilentool „*installutil.exe*“ können Services installiert werden. Dies ist auf Windows Server bereits vorinstalliert.
Bei der Installation wird ein Dienst in einer Registrierungsdatenbank eingetragen. Der Service selbst kann als ausführbare .EXE Datei vorliegen, oder auch als Dynamic Link Library (.DLL). Liegt der Service als .DLL vor, führt die ausführbare Datei „*Svchost.exe*“ (service host) den Service aus der DLL aus.

2.2.2 Beispielsklasse Serviceimplementierung

Um einen Service zu implementieren, muss eine Klasse erstellt werden, welche von der Klasse ServiceBase ableitet. Dazu muss die Assembly System.ServiceProcess als Referenz dem Projekt hinzugefügt werden. Nun können in der Service-Klasse die verschiedenen On-Methoden überschrieben werden, um auf bestimmte Events zu reagieren.

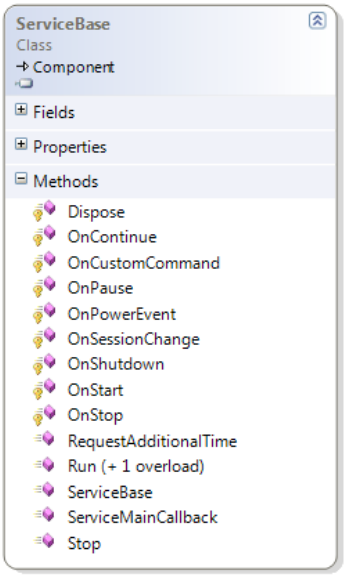


Abbildung 14: ServiceBase Klasse aus System.ServiceProcess

2.2.3 Beispielsklasse Serviceinstallation

Damit der Service mit dem “*installutil.exe*” Tool installiert werden kann, muss eine Installationsklasse mit dem RunInstaller Attribut erstellt werden:

```
using System.ComponentModel;
using System.Configuration.Install;
using System.ServiceProcess;

namespace ServiceExample
{
```

```
    [RunInstaller(true)]
    public class ServiceExampleInstaller : Installer
    {
        private ServiceInstaller _service;
        private ServiceProcessInstaller _serviceProcess;

        public ServiceExampleInstaller()
        {
            _service = new ServiceInstaller();
            _serviceProcess = new ServiceProcessInstaller();

            _serviceProcess.Account = ServiceAccount.NetworkService;
            _service.ServiceName = "Service Example";
            _service.StartType = ServiceStartMode.Manual;
            Installers.Add(_service);
            Installers.Add(_serviceProcess);
        }
    }
}
```

"installutil.exe" sucht sich via Reflection die Klasse aus dem Assembly mit dem Attribut RunInstaller und führt diese dann aus.

2.2.4 Umsetzung im Projekt

Die Idee beim Network Testcase Engine Service ist es nun, in der OnStart() Methode eine .NET Remoting Schnittstelle im Netzwerk bekannt zu machen. Diese Schnittstelle wird vom Webinterface konsumiert, welches dem Benutzer das GUI bereitstellt. Der Service ist nötig, da eine Webseite nur kurzlebig ist (Dauer einer Anfrage) und somit die virtuellen Maschinen nicht verwalten kann. Der Service hingegen läuft im Hintergrund und startet auf Befehl vom Webinterface zum Beispiel die virtuellen Maschinen „load balanced" auf. Das heisst, dass die VMs auf die verfügbaren Nodes aufgeteilt werden und pro Computer, abhängig von der Anzahl Cores, nur eine bestimmte Anzahl gleichzeitig aufgestartet werden. Gemeinsame Daten werden in einer Datenbank, welche beiden Komponenten bekannt ist abgespeichert.

2.3 ASP.NET MVC

ASP.NET MVC ist ein Web Framework von Microsoft. Wie der Name sagt, baut es auf das Model-View-Controller Pattern auf. Dieses Pattern wird oft in der Softwareentwicklung eingesetzt um die Daten, Präsentation und Steuerung voneinander zu trennen. Die Trennung bringt mehrere Vorteile mit sich. Zum Beispiel können die drei Komponenten unabhängig und getrennt voneinander mit Unit Tests getestet werden. Der Aufbau und die Struktur werden durch das MVC klar vorgegeben. Die Kohäsion innerhalb der Komponenten wird dadurch erhöht und die Abhängigkeiten verringert.
Auf ASP.NET angewendet beinhalten die einzelnen Komponenten des Patterns folgendes:

Model: Business Objekte, welche vom Controller aufbereitet und in der View dargestellt werden.

View: Zu den Views gehört HTML Code mit zusätzlichen ASP.NET Anweisungen. Diese Anweisungen werden vom Framework intepretiert und umgesetzt. Mit den Views wird die Funktionalität der Applikation für den Endbenutzer zugänglich gemacht. Der Benutzer interagiert nie direkt mit den Business Objekten, er erreicht diese über Eingabemasken, welche von den Views ihm bereitgestellt werden.

Controller: Der Controller nimmt die Interaktionen vom Benutzer entgegen und reagiert entsprechend. Er kann zum Beispiel beim Zugriff auf eine Seite, auf welche der Benutzer keinen Zugriff hat, den Benutzer auf eine andere Seite umleiten.

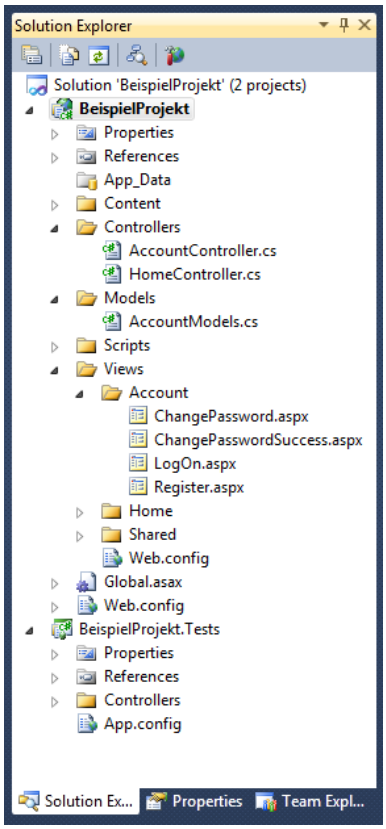


Figure 1: Aufbau eines ASP.NET MVC Projektes

2.3.1 Umsetzung im Projekt

Bei der Network Testcase Engine (NTE) wird das Webinterface mittels dem ASP.NET MVC Framework umgesetzt. ASP selber bietet eine Benutzerverwaltung von Haus aus. Jedoch stellte sich diese für das NTE Projekt als zu umfrangreich aus. Deswegen wird eine eigene Benutzerverwaltung implementiert, welche mit einer Benutzertabelle auskommt.
Das Model wird mit dem Code-First Ansatz des Entity Framework gemapped. Da das Webinterface nicht die einzige Applikation im System ist, welche auf die Datenbank zugreifft, befindet sich das Model in einem eigenen Namespace und wird als DLL deployed.

2.4 Interprozess-Kommunikation (.NET Remoting)

.NET Remoting ist ein umfassendes Framework zur nahtlosen Kommunikation zwischen Objekten über die eigene Applikationsdomäne/Prozessgrenze hinweg. Dabei können sich die Prozesse auf verschiedenen Computern befinden. Mit Referenzen auf Objekte kann „normal“ gearbeitet werden, als ob sich die Objekte im gleichen Prozessraum befinden würden. Beim Auflösen der Referenzen wird durch ein generiertes Proxyobjekt automatisch geprüft, ob das Objekt zu einem anderen Prozess gehört oder ob der Aufruf erst über das Netzwerk geschickt werden muss. Solche „Remoting“-fähige Objekte müssen lediglich von MarshalByRefObject abgeleitet sein:

```
class VM : MarshalByRefObject
{
    public string IP { get; set; }
    public string Hostname { get; set; }
    public int Port { get; set; }
}
```

2.4.1 Service bereitstellen

Ein Service kann mit wenigen Codezeilen für andere bereitgestellt werden. Es werden 3 Schritte dazu benötigt:

1. Registrierung des Services mit Typ, Name und wie bekannte Objekte aktiviert werden
2. Einen `TCPChannel` erstellen (auf bestimmten Port)
3. Den Channel bei `ChannelServices` registrieren

Da die Einstellungen aus Punkt 1 nicht ändern, kann dieser Schritt im static Block einer Klasse durchgeführt werden. Dieser wird ausgeführt, wenn eine Klasse von der Runtime geladen wird.
Die Erstellung des Channels und die Registrierung wird üblicherweise in einer `Start()` Methode durchgeführt.
Folgender Codeausschnitt zeigt ein Beispiel, wie das Interface `IReport` mit .NET Remoting öffentlich gemacht werden kann:

```
class ReportingServer : IReport
{
    private int _port;
    private TcpChannel _tcpChannel;

    static ReportingServer()
    {
        RemotingConfiguration.RegisterWellKnownServiceType(
            typeof(IReport), "ReportingServer", WellKnownObjectMode.SingleCall);
    }

    public ReportingServer(int port)
    {
        _port = port;
    }

    public void Start()
    {
        _tcpChannel = new TcpChannel(_port);
        ChannelServices.RegisterChannel(_tcpChannel, false);
    }

    public void Stop()
    {
        ChannelServices.UnregisterChannel(_tcpChannel);
    }

    public void VMRemoteControlDebug(string ip, string message)
    {
    }

    public void LoadGeneratorDebug(string ip, string message)
    {
```

```
        }
    }
```

3. Analyse der Tasks

Tasks sind Bausteine, welche in einer gewünschter Reihenfolge in einem Testcase konfiguriert werden können. Zu einem Task gehören Eigenschaften mit einem Namen und einem Wert. In diesem Kapitel werden die Tasks analysiert und die benötigten Eigenschaften festgelegt.

3.1 MPPAction

Der MPPAction Task kann verschiedene Aktionen gegen das MPP ausführen.

Aktion	Beschreibung / verwendete Eigenschaften
TestRedirect	Surft eine Seite an und überprüft, ob das MPP auf die Loginseite umleitet.
TestLoggedIn	Überprüft ob man eingeloggt ist, indem URL besucht wird. Falls man auf der Landing-Page landet, schlägt diese Aktion fehl.
Login	URL wird besucht, um vom MPP auf die Landing-Page umgeleitet zu werden. Das Formular wird dann ausgefüllt und abgeschickt. Login Password
Logout	Meldet die aktuelle Session beim MPP ab.

Tabelle 2: Task MPPAction

3.2 DNSResolver

Der DNSResolver Task löst eine übergebene DNS in eine IP auf.

Aktion	Beschreibung / verwendete Eigenschaften
Resolve	Versucht ein Domainname aufzulösen. DN MaxRetries

Tabelle 3: Task DNSResolver

3.3 Wait

Der Wait Task wartet entweder eine vorgegebene Zeit, oder zufällig in einem vorgegebenen Wertebereich.

Aktion	Beschreibung / verwendete Eigenschaften
Wait	Wartet für in Time angegebene Anzahl an Millisekunden. Time
RandomWait	Wartet zufällig in einem vorgebenen Zeitraum Min Max

Tabelle 4: Task Wait

3.4 Ping

Mit dem Ping Task können Pingbefehle ausgeführt werden.

Aktion	Beschreibung / verwendete Eigenschaften
Ping	Pingt die angegebene IP an. Wenn der Ping nicht beantwortet wird, versucht es der Task erneut, bis MaxRetries erreicht wurde. IP MaxRetries

Tabelle 5: Task Ping

3.5 Googler

Mit dem Googler Task kann ein Benutzer simuliert werden, welcher verschiedene Suchbegriffe bei Google eingibt und ein paar Ergebnisse besucht.

Aktion	Beschreibung / verwendete Eigenschaften
Surf	Simuliert eine Person, die auf Google nach bestimmten Schlüsselwörter sucht. Über die Properties können die Keywords, die Anzahl an Suchanfragen, die Anzahl an zu besuchenden Hits und wie tief in den Ergebnissen weiter Links verfolgt werden, definiert werden. Keywords NumberOfSearches NumberOfHitsToVisit

Table 1: Task Googler

4. Anforderungsspezifikation

4.1 Allgemeine Beschreibung

4.1.1 Produkt Perspektive

Als Folgearbeit des Captive Portal Load Generator, soll nun eine Lösung entwickelt werden, welche dem Unit Testing bei der Softwareentwicklung ähnelt. Verschiedene Testcases sollen wie mit einem Baukasten zusammengestellt werden können. Dabei werden Tasks workflowartig für jeden Testcase definiert. Ein Task ist zum Beispiel eine Webseite zu besuchen oder beim MPP anmelden.
Ferner soll es möglich sein, ein durchgeführter Testcase nochmals mit den selben Laufzeiteinstellungen (wie zum Beispiel zufällig generierte Wartezeiten) in einem Replayer durchzuführen.
Die ganze Applikation soll über eine Website gesteuert werden können, welche stateless ist. Das System soll unabhängig vom Client gesteuert werden können.

4.1.2 Produkt Funktion

Folgende Funktionen werden gefordert:

- Verwaltung von TestCases
    - Erstellen
    - Editieren
    - Löschen
- Starten und Stoppen eines TestCases
- Wiederholtes ausführen des TestCases nach beendigung mit gleicher Initialisierung
- Statistiken ansehen von abgeschlossenen TestCases
- Verteilung der Komponenten auf die verschiedenen Computer soll automatisch geschehen

4.1.3 Benutzercharakteristik

Die Applikation richtet sich an erfahrene Benutzer, welche mit dem MPP und dem Captive Portal Load Generator vertraut sind. Für die Entwicklung von weiteren Tasks sind nur Kenntnisse über die Task API notwendig. Um die Architektur müssen sich Taskimplementierungen nicht kümmern.

4.1.4 Einschränkungen

Um die Applikation zu benützen, wird ein Webbrowser mit Javascriptunterstützung vorausgesetzt.

4.1.5 Annahmen

Keine

4.1.6 Abhängigkeiten

Die Applikation ist vom MPP selber unabhängig. Der bestehende Quellcode vom Captive Portal Load Generator wird übernommen und gegenfalls angepasst.

4.2 Spezifische Anfoderungen

Die Anforderungen werden nach dem FURPS+-Modell kategorisiert. Diese Qualitätsmerkmale dienen als Checkliste für die Behandlung von Anforderungen, um das Risiko zu verringern, eine wichtige Facette des Systems zu übersehen.

4.2.1 Funktionalität

4.2.2 Erweiterbarkeit
Die Applikation soll erweiterbar sein. Als Beispiel für eine Erweiterung käme das Hinzufügen von weiteren Tasks in Frage.

4.2.3 Bedienbarkeit
Die Bedienung des Webinterfaces soll einfach und schnell möglich sein.

4.2.4 Erlernbarkeit
Die Einarbeitungszeit in die Applikation soll lediglich wenige Minuten in Anspruch nehmen. GUI Elemente sind intuitiv zu platzieren und zu beschriften.

4.2.5 Wiederherstellbarkeit
Bei einem Programmabsturz dürfen die Laufzeitdaten des aktuellen TestCases nicht verloren gehen.

4.2.6 Fehlerbehandlung
Die Anwendung soll den Benutzer vor Fehler schützen. Der Benutzer soll von der Konfiguration bis zum Ende eines Testcasedurchlauf klar geführt werden.

4.2.7 Zuverlässigkeit
Abstürze dürfen druch die Applikation selbst kein entstehen. Externe Faktoren, welche die Applikation zum Abstürzen bringen, können natürlich nicht ausgeschlossen werden.

4.2.8 Leistung
An die Leistung der Webapplikation sind keine speziellen Anforderungen gestellt. Sie wird über das Web erreicht. Daher sollte diese nicht überladen sein um die Seitenladezeit unnötig zu vergrössern.

4.2.9 Wartbarkeit
Der Code soll gut strukturiert abgelegt werden. Dabei gehören logische Gruppen von Klassen und Typen in gemeinsame Namespaces. Zirkuläre Abhängigkeiten sind verboten.

4.3 Schnittstellen

4.3.1 Benutzerschnittstelle(UI)
Die Anwendung ist hauptäschlich über die Maus im Webbrowser steuerbar.

4.3.2 Softwareschnittstelle
- Programmiersprache
    - Microsoft C#.NET 4.0
    - Microsoft C#.NET 3.5
    - Microsoft ASP.NET 4.0

- User Interface
    - ASP.NET
- Datenkbank
    - Microsoft SQL Server 2008
- Windows Services
- MPP
- Zugriff via http

4.4 Lizenzanforderungen

Es gelten keine speziellen Lizenzanforderungen.

4.5 Verwendete Standards

Im gesamten Projekt wird nach den Microsoft Coderichtlinien programmiert.

4.6 User Stories

Folgende User Stories wurden in der Analysephase im TFS erfasst. Im Kapitel Aufgaben werden alle Aufgaben aufgelistet.

3	**Benutzer kann Test Cases im Webinterface verwalten**	**ymyr**	**Captive\Iteration 02**
	Benutzer kann Test Cases über Webinterface verwalten. Dabei können die Tasks Workflowmässig in eine Reihenfolge gebracht werden. Tasks werden in Kästchen angeordner. Innerhalb von Kästchen kann in einem ersten Dropdown Menu die Methode ausgewählt werden. Direkt darunter werden die benötigten Properties für diese Methoden angezeigt, für welche einen Wert spezifiziert werden kann. Am Ende vom Kästchen kann einen Expected Value Wert angegeben werden. => Checkbox ob wirklich erwünscht.		

4	**Benutzer kann im Webinterface ein Szenario starten**	**ymyr**	**Captive\Iteration 03**
	Der Benutzer wählt ein Szenario aus und erhält nochmals alle Details zum Szenario aufgelistet. Wenn die Details OK. sind, können über den Button Node Discovery alle Nodes im Netzwerk aktiviert werden. Wenn genügend Resourcen (Nodes welche VMs hosten) verfügbar sind, erscheint ein Button Start, um das Szenario zu starten.		

5	**Benutzer kann im Webinterface ein durchgeführten Testcase wiederholen**	**oli**	**Captive\Iteration 06**
	Fertig durchgeführter Test Case kann wiederholt werden. Dabei ist es wichtig, dass die selben Einstellungen verwendet werden (verwendete Zufallswartezeiten etc.)		

12	**Benutzer kann Reports von durchgelaufenen Testcases anschauen.**	**ymyr**	**Captive\Iteration 05**
	Auf einer Übersichtsseite werden alle Reports von durchgelaufenen Szenarien aufgelistet. Neben dem Report Namen soll ein Indikator sein, ob alles OK verlaufen ist oder nicht. Über die Liste können die Details der Reports aufgerufen werden. Auf der Detailseite eines Reports wird nach den Testcase Typen gruppiert und jeweils alle Tasks vom jeweiligen Testcase aufgelistet. Zu jedem Task steht wie oft er fehlgeschlagen ist (mit Prozentanzeige) und die benötigte Durchschnittszeit für den Task. Am Ende des Reports wäre noch eine graphische Auswertung möglich, welche aber noch nicht spezifiziert ist.		

15	**Benutzer kann Benutzer für das Webinterface verwalten**	**ymyr**	**Captive\Iteration 01**
	Um Zugriff auf das Webinterface zu erhalten, muss sich der Benutzer authentifizieren. Folgende Eigenschaften werden für jeden Benutzer gespeichert: Login, Passwort (SHA1), E-Mail. Benutzer sollen über eine Seite verwaltet werden können (CRUD).		

18	**Benutzer kann verfügbare Tasks im Webinterface registrieren**	**ymyr**	**Captive\Iteration 04**
	Damit das Webinterface die möglichen Tasks, welche in Testcases platziert werden können, kennt, müssen diese registriert werden. Die Verwaltung der verfügbaren Tasks soll über eine Seite im Webinterface gehen. Gespeichert werden diese in einer Tabellen mit folgenden Spalten: ID, Typ. Um die Verwaltung zu vereinfachen, soll ein Verzeichnis automatisch nach den Task DLLs abgesucht werden und via Reflection die Klassen aus den DLLs lesen. Differenzen der Tabelle sollen per Knopfdruck korrigiert werden können --> es soll kein manuelles Tippen der Tasktypen nötig sein.		

188	**LoadGenerator kann Testcase vom NTE Service entgegennehmen und ausführen**	**oli**	**Captive\Iteration 01**
	DTO Klasse implementieren, welche die nötigen Informationen speichert. Remoting Methode zur Übergabe des Objektes implementieren.		

189	**Reporting Service für allgemeines Logging**	**oli**	**Captive\Iteration 01**
	Komponenten können dem Reporting Service Debugging Informationen schicken		

190	**Benutzer kann Ping Task in Testcase einsetzen**	**oli**	**Captive\Iteration 02**
	Ping Task implementieren und die nötigen Annotations an die Methoden und Properties schreiben, damit die Webapplikation den Task erkennt.		

191	**Benutzer kann Wait Task in Testcase einsetzen**	**oli**	**Captive\Iteration 01**
	Mit dem Wait Task kann in einem Testcase entweder eine vordefinierte Zeit gewartet werden, bis mit dem nächsten Task fortgefahren wird, oder in einem vorgegeben Zeitraum für eine zufällige Zeit.		

192	**Benutzer kann DNSResolver Task in Testcase einsetzen**	**oli**	**Captive\Iteration 01**
	Dem DNSResolver Task kann ein DN übergeben werden, welcher dann aufgelöst wird.		

193	**Benutzer kann MPPAction Task in Testcase einsetzen**	**oli**	**Captive\Iteration 01**
	Mit dem Task MPPAction kann man sich beim MPP ein-/ausloggen oder überprüfen, ob man auf die Landing Page umgeleitet wird.		

194	**Der Benutzer hat ein Mini-Framework, um weitere Tasks zu erstellen**	**oli**	**Captive\Iteration 06**
	Wenn in Zukunft neue Ideen für Tasks aufkommen, sollen diese möglichst einfach integrierbar sein. Der Task selbst soll nichts über die Architektur des gesamten Systemes wissen, er muss lediglich von einer vorgegebenen Klasse ableiten und die Interfaces kennen, um Ereignise weiter zu geben.		

202	**Benutzer kann Node Discovery durchführen**	**oli**	**Captive\Iteration 02**
	Bevor ein Szenario gestartet werden kann, müssen alle Nodes im Netzwerk dem NTE.Service bekannt sein, damit der Service virtuelle Maschinen auf die Nodes verteilen kann. Im Webinterface soll über ein Knopf ein Multicast Paket ins Netzwerk geschickt werden können, auf welches sich alle Nodes im Netzwerk melden. Dadurch entfällt eine manuelle Konfiguration jeder einzelner Node.		

251	**Benutzer kann Googler Tast in Testcase einsetzen**	**oli**	**Captive\Iteration 06**
	Der Googler Task bekommt eine Liste von Suchwörtern, aus welchen er zufällig ein paar auswählt und danach sucht.		

252	**Benutzer kann MPPHammer Task in Testcase einsetzen**	**oli**	**Captive\Iteration 03**
	Der MPPHammer Task surft verschiedene Seiten aus einer Liste an, ohne sich dabei beim MPP einzuloggen. Er überprüft nach jedem Aufruf, ob er auf die Landing Page weitergeleitet wurde.		

4.7 Aufgaben

Alle Aufgaben während der Entwicklungsphase wurden im TFS als Workitem mit dem Typ Task erfasst. Alle Tasks sind mit der dazugehörigen User Story verknüpft. So kann einfach überprüft werden, was für eine bestimmte User Story noch erledigt werden muss, um sie abzuschliessen.

187	**Szenario Verwaltungsseite erstellen**	**ymyr**	**Captive\Iteration 01**
	Eine übersichtliche Seite mit allen Szenarioen aufgelistet. Nebst dem Szenario Name soll auch aufgelistet werden, wann das Szenario das letztemal durchgeführt wurde.		

195	**Task Verwaltungsseite erstellen**	**ymyr**	**Captive\Iteration 05**
	Via Reflection DLLs auslesen und mit Tabelle abgleichen. Die Attribute sind im BA Dokument dokumentiert.		

196	**Management Tabelle für verfügbare Tasks erstellen**	**ymyr**	**Captive\Iteration 05**
	Die Tabelle enthält folgende Felder: ID (int), Type (string).		

197	**Attribute von Tasks Dokumentieren**	**oli**	**Captive\Iteration 02**
	Im BA Dokument die Attribute Dokumentieren, damit auch andere Entwickler später neue Tasks für die NTE schreiben können.		

198	**JS für einfachere Konfiguration**	**ymyr**	**Captive\Iteration 03**
	Die Testcases sollen möglichst mit der Maus zusammen geklickt werden können. Siehe dazu GUI Prototyping.		

199	**Autzentifizierung**	**ymyr**	**Captive\Iteration 01**
	Der Benutzer kann sich auf der Webseite mittels Benutzernamen und Password einloggen und auch wieder ausloggen		

201	**Benutzerverwaltungsseite**	**ymyr**	**Captive\Iteration 01**
	Der Benutzer hat eine Übersicht über die aktuellen Benutzer in der Datenbank. Aus dieser Übersicht können die Benutzer verwaltet (CRUD) werden.		

204	**Runtime Eigenschaften zwischenspeichern**	**oli**	**Captive\Iteration 06**
	Damit später Testcase Runs wiederholt werden können, müssen Runtime-Eigenschaften gespeichert werden. Diese werden dann beim wiederholen geladen.		

207	**Design von Tasks dokumentieren**	**oli**	**Captive\Iteration 02**
	Im BA Dokument muss die API der Tasks dokumentiert sein. Ein neuer Entwickler soll nur mit dem Wissen der API Task bereits neue Tasks entwickeln können. Dabei spielt die ganze Architektur hinter dem NTE keine Rolle.		

211	**Ping Task implementieren**	**oli**	**Captive\Iteration 01**

	-

212	**Wait Task implementieren**	**oli**	**Captive\Iteration 01**
	SpecificWait: eine übergebene Anzahl an Millisekunden warten RandomWait: der Task wartet zufällig in einer vorgegebenen Range von Millisekunden		

213	**DNSResolver Task implementieren**	**oli**	**Captive\Iteration 01**
	Angegebener Domainname wird aufgelöst		

214	**MPPAction Task implementieren**	**oli**	**Captive\Iteration 01**
	In dieser Assembly werden alle Aktionen gegen das MPP implementiert **Login:** surft www.kernel.org an um MPP URL herauszufinden und loggt sich anschliessend beim MPP ein **Logout:** Loggt sich beim MPP aus **TestLoggedIn:** Überprüft ob der Client eingeloggt ist **TestRedirect:** Überprüft ob das MPP den Client auf die Landingpage weiterleitet		

234	**Report Webseite erstellen**	**ymyr**	**Captive\Iteration 05**
	Seite mit Tabelle, in welcher die wichtigsten Daten zu einem Szenariodurchlauf aufgelistet werden: Szenario Name, Startdatum, Enddatum, Benutzer, Kurzergebnis (OK oder fehler vorhanden). Mit einem Klick auf einen Tabelleneintrag öffnet sich ein Popup mit einer Detailansicht zum Szenariodurchlauf. Hier sollen alle Task Resultate aufgelistet werden, gruppiert nach VMs.		

235	**Basic Logging für Nodes und LoadGenerators**	**oli**	**Captive\Iteration 01**
	Der Output der VMRemoteControl und LoadGenerator Anwendung soll wenn gewünscht gelogged werden.		

236	**Schnittstelle implementieren**	**oli**	**Captive\Iteration 01**
	Die Schnittstelle ist im SAD dokumentiert.		

237	**Ausführung von übergebenem Testcase auf VM**	**oli**	**Captive\Iteration 02**
	In der LoadGenerator den übergebenen Task ausführen.		

238	**Logging auf Webseite einbetten**	**ymyr**	**Captive\Iteration 05**
	Die Logs sollen übersichtlich auf der Webapplikation aufgelistet werden. Um Logeinträge schneller zu finden soll eine Ajax-Suche vorhanden sein.		

239	**Basisklasse für Tasks erstellen**	**oli**	**Captive\Iteration 01**
	Basisklasse für alle Tasks erstellen. Sie stellt eine Methode zur Verfügung, damit der Task auf der Shell etwas ausführen kann.		

240	**Informationen an Reporting/Service senden**	**oli**	**Captive\Iteration 02**

	Die Schnittstelle ist im SAD dokumentiert.		

241	**Webinterface Grundgerüst aufbauen**	**ymyr**	**Captive\Iteration 01**
	Die Grundstruktur der Webapplikation mit ExtJS aufbauen. Hilfsklasse für die Serialisierung der Modelklassen in das JSON Format sind nötig.		

243	**Benutzerübersicht**	**ymyr**	**Captive\Iteration 01**
	Es werden alle Benutzer aus der Datenbank in einer Tabelle dargestellt		

244	**Neuen Benutzer erstellen**	**ymyr**	**Captive\Iteration 01**
	Es kann mittels einem Formular ein neuer Benutzer erstellt werden mit folgenden Details: Login, Password (SHA1), Email Die Login Spalte ist unique.		

245	**Benutzer editieren**	**ymyr**	**Captive\Iteration 01**
	Ein selektierter Benutzer aus der Übersichtstabelle kann in einem Formular editiert werden. Das Password ist kein zwingendes Feld zum ausfüllen beim editieren		

246	**Benutzer löschen**	**ymyr**	**Captive\Iteration 01**
	Ein selektierter Benutzer in der Übersicht kann per Knopfdruck gelöscht werden.		

247	**Formular für das Kreieren/Editieren eines Benutzers**	**ymyr**	**Captive\Iteration 01**
	Ein Formular erstellen welches die Eigenschaften (Login, Password, Email) eines Benutzers beinhaltet. Validierung der Felder: Login : alphanumerisch Email: email, z.B. user@example.com		

248	**JS Validierung Benutzername**	**ymyr**	**Captive\Iteration 01**
	Mittels Javascript Live überprüfen ob der gwählte Benutzername verfügbar ist		

249	**Login**	**ymyr**	**Captive\Iteration 01**
	Mittels einem Formular welches die Feler Login und Email zum eingeben hat loggt sich der Benutzer ein.		

250	**Logout**	**ymyr**	**Captive\Iteration 01**
	Auf der Hauptansicht der Webseite kann sich der Benutzer über einen Button aus der Webseite ausloggen.		

253	**Googler Task implementieren**	**oli**	**Captive\Iteration 06**
	3 Properties müssen gesetzt werden können: Keywords, NumberOfSearches, NumberOfHitsToVisit		

256	**OR Mapping mit EF4**	**oli**	**Captive\Iteration 01**
	Mapping von NTE.Model Klassen auf Tabellen. Code-First Ansatz.		

258	**Node Discovery implementieren**	**oli**	**Captive\Iteration 02**
	Nodes per Multicast Paket mit ASCII Inhalt „request_register“ triggern, ihre Informationen über Hardware und .NET Remoting Schnittstelle dem Service zu schicken.		

259	**Webseite zur Konfiguration des Durchlaufs erstellen**	**ymyr**	**Captive\Iteration 02**
	Wie im GUI Prototyping bereits vorgeschlagen, die Funktionalität auf die Webseite bringen. Dabei soll das ExtJS Framework eingesetzt werden, um live Daten anzuzeigen. Der Ablauf ist in der verlinkten User Story definiert.		

292	**Verteilung der VMs auf die Nodes**	**oli**	**Captive\Iteration 03**
	Der NTE.Service muss die Anzahl der benötigten VMs für ein Szenario auf die vorhandenen Nodes verteilen.		

344	**Task via Reflection ausführen**	**oli**	**Captive\Iteration 04**
	Die LoadGenerator Anwendung bekommt ein Testcase Objekt vom NTE Service. Die einzelnen Tasks von diesem Testcase müssen via Reflection ausgeführt werden, da die Tasks nicht statisch gebunden werden.		

345	**Reihenfolge ändern**	**ymyr**	**Captive\Iteration 06**
	Im Testcase Editor soll die Reihenfolge der Tasks per Drag and Drop verändert werden können.		

346	**Task mit Drag&Drop einfügen**	**ymyr**	**Captive\Iteration 06**
	Die Eigenschaften/Methoden können gemäss Task API anhand von vordefinierten Attributen ausgelesen werden.		

347	**Testcase Konfigurationsseite erstellen**	**ymyr**	**Captive\Iteration 06**
	Workflowartiger Editor mit Javascript umsetzen.		

367	**MAC Generator / Verteilung der VMs**	**oli**	**Captive\Iteration 03**
	MAC Adressen müssen synchronisiert generiert werden, damit keine doppelten MAC Adressen im Netzwerk vorhanden sind. Die VMs werden nach Kapazität der einzelnen Nodes verteilt.		

368	**Reflection Logik in Basisklasse für Tasks implementieren**	**oli**	**Captive\Iteration 04**
	Bei der Implementation eines Tasks soll kein Architekturcode nötig sein. Dies soll die Basisklasse übernehmen, um eine hohe Kohäsion zu erhalten und das entwickeln eines Tasks zu vereinfachen.		

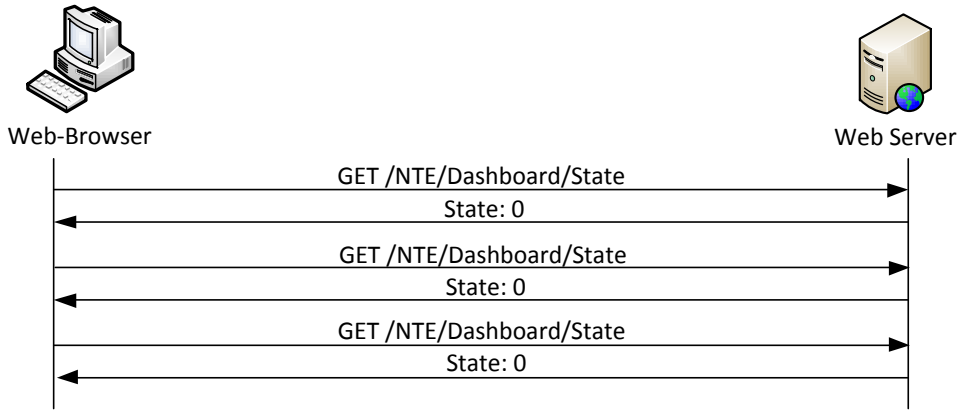
369	**Backend implementieren zum entgegennehmen der Ergebnisse**	**oli**	**Captive\Iteration 05**
	Die Requests kommen asynchron rein und müssen entsprechend synchronisiert werden.		

5. Web Frontend

Gemäss Anforderungen muss das Web Frontend stateless sein. Dadurch erhält der Benutzer viel Flexibilität, welche beim Captive Portal Load Generator nicht vorhanden war. Oft muss lange gewartet werden, bis eine weitere Benutzerinteraktion notwendig ist. Durch das stateless Design kann der Benutzer die Webseite schliessen und später, wenn eine Benutzerinteraktion notwendig ist, neu aufmachen um die Interaktion zu tätigen.
Damit dies möglich ist, muss der Service im Hintergrund den State speichern. Da mehrere Benutzer gleichzeitig das Web Frontend benützen können, und somit der State zu jeder Zeit geändert werden könnte, müssen die Clients fähig sein, solche Änderungen zu bermerken und darauf zu reagieren.
Mit Ajax können asynchrone http Requests gemacht werden, um ein solches Verhalten zu implementieren. Es existieren grundsätzlich 2 Techniken, um im Client auf Updates zu reagieren: Polling und Comet (Push).

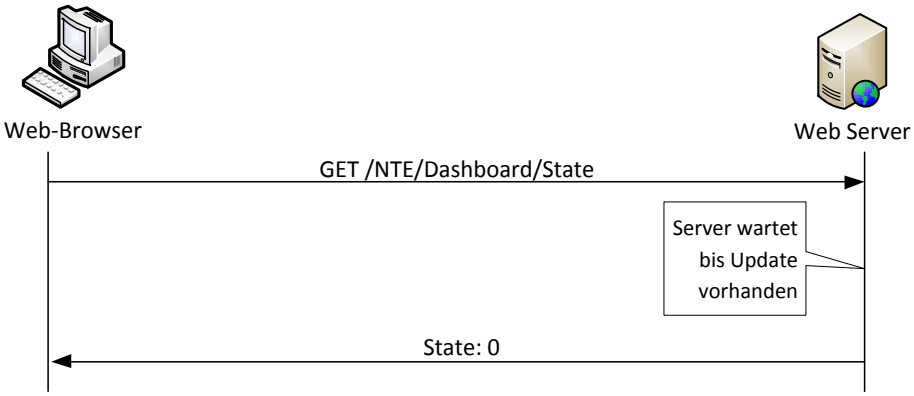
5.1 Polling

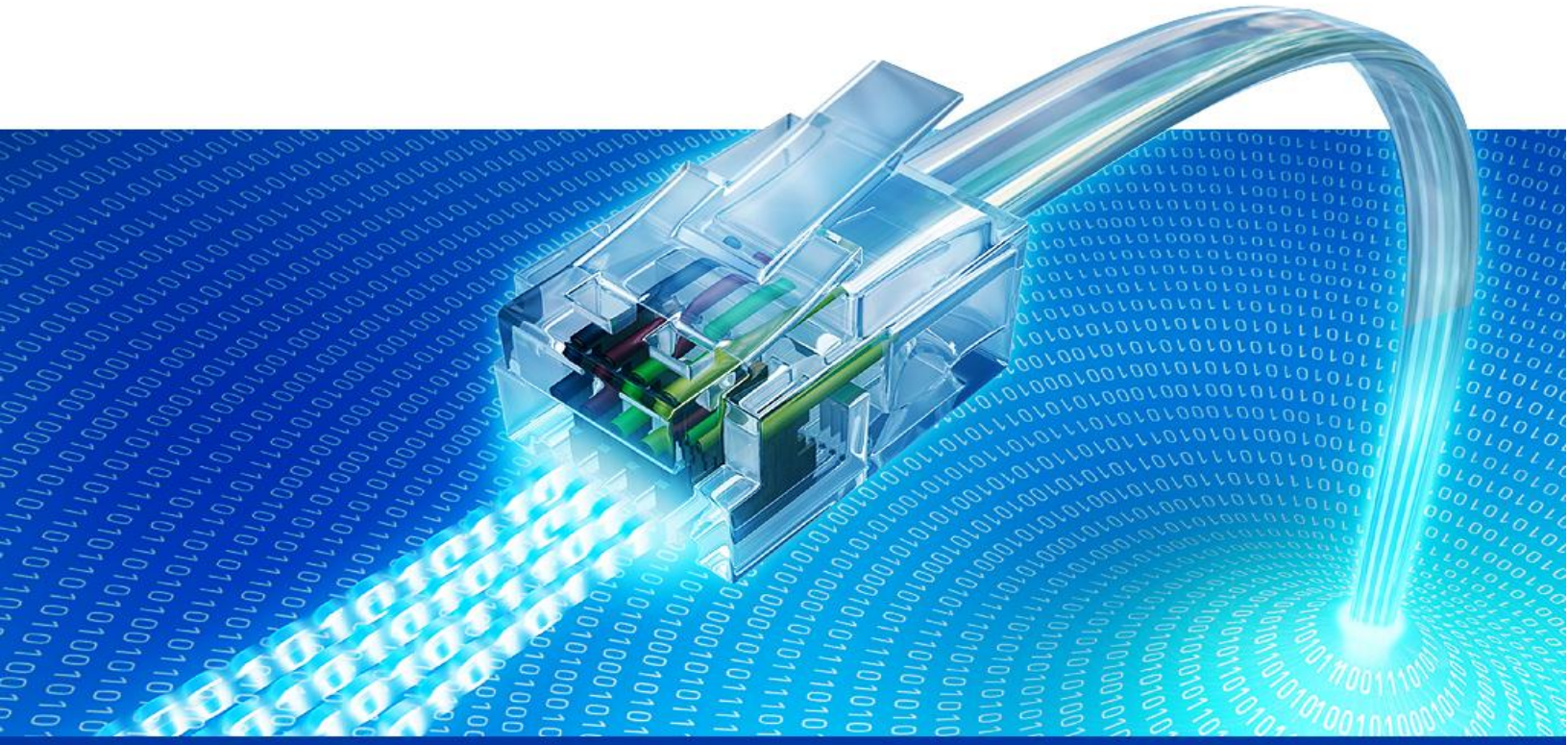
Beim Polling sendet der Client in einem festgelegten Zeitintervall Requests an den Server und überepruft so nach Updates. Diese Variante ist schnell umgesetzt, bringt aber einige Nachteile mit sich. Je nach Zeitintervall sind die Updates immer leicht verzögert. Der Client wird durch die vielen Requests im Hintergrund belastet. Serverseitig ist das Problem mit der Belastung viel verherender. Eine Webseite, welche Polling einsetzt, skaliert schlichtweg extrem schlecht.



5.2 Comet (Push)

Das Comet Prinzip löst die Probleme von Polling. Es wird ein Ajax Request an den Server geschickt, der Server antwortet aber erst, wenn ein Update vorliegt. Dadurch muss der Server weniger Requests verarbeiten. Dennoch muss er unter Umständen viele Verbindungen offen halten können.





8. Software Architcture Document

Network Testcase Engine

Bachelorarbeit FS 2010

Technische Hochschule Rapperswil

1. Komponenten

Das Network Testcase Engine Projekt umfasst mehrere Komponenten, welche auf mehrere Computer verteilt werden. In diesem Kapitel werden die Komponenten kurz beschrieben und die Schnittstellen definiert.

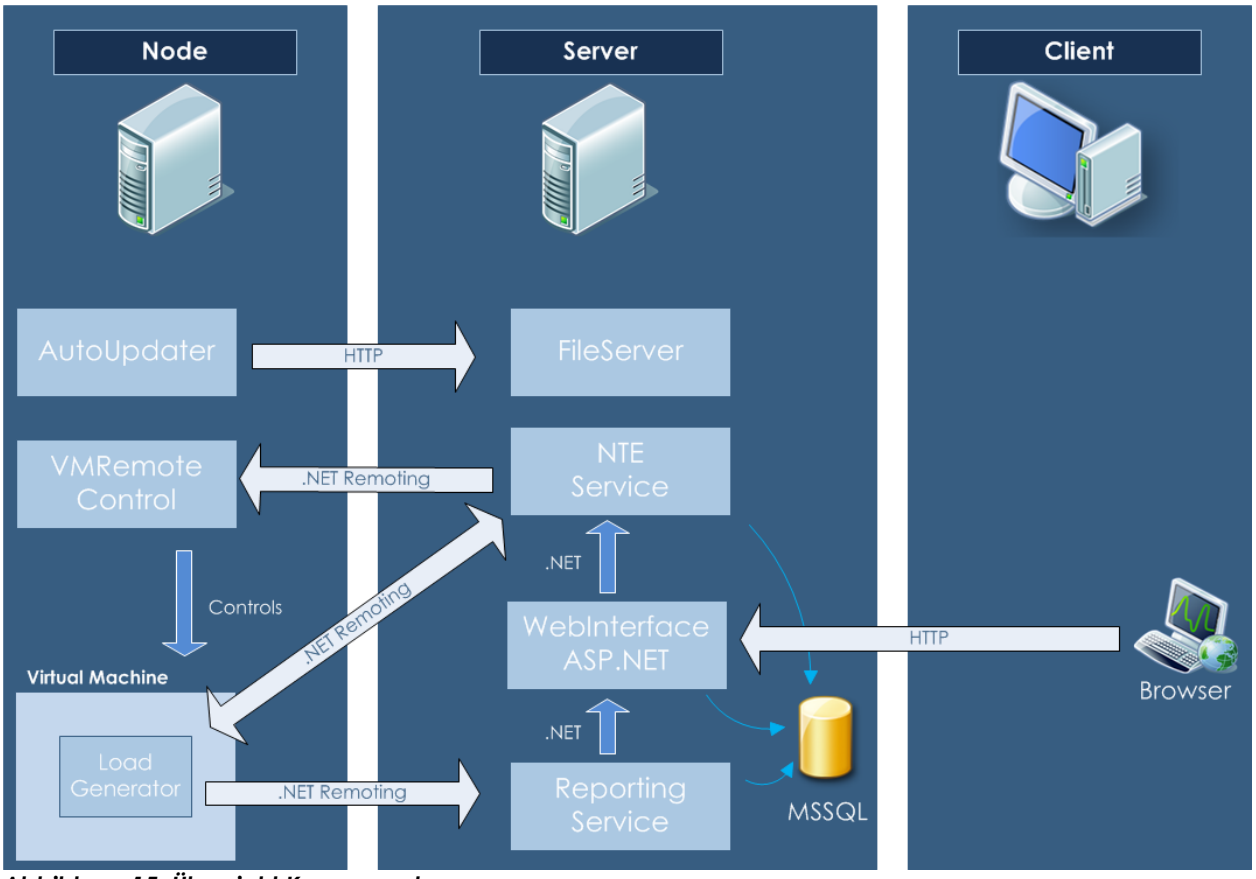


Abbildung 15: Übersicht Komponenten

1.1 AutoUpdater

Wenn Änderungen am Projekt gemacht werden, müssen diese oft auf verschiedene Computer übertragen werden. Um diesen Prozess zu automatisieren, dient die AutoUpdater Anwendung. In einer XML-Datei kann konfiguriert werden, welche Dateien auf Updates überprüft werden sollen. Diese Komponente ist nur für die Entwicklung relevant. Sie ermöglicht es, kleine Änderungen beim Testen nach dem Kompilieren innert Sekunden auf alle Computer zu verteilen.

1.2 ChecksumGenerator

Diese Anwendung generiert eine XML Datei, welche die MD5 Hashes aller Dateien in einem bestimmten Ordner beinhaltet. Diese Datei wird vom AutoUpdater bezogen, um die lokalen Dateien mittels Checksummenvergleich auf Änderungen zu überprüfen.

1.3 VMRemoteControl

Die VMRemoteControl Anwendung läuft auf allen Rechner („Nodes“), welche ihre Hardware zur Virtualisierung zur Verfügung stellen. Sie bietet gegen aussen eine .NET Remoting Schnittstelle zur Fernsteuerung der virtuellen Maschinen.

Abbildung 16: Interface IVMControl

1.4 LoadGenerator

Die LoadGenerator Anwendung wird innerhalb der virtuellen Maschinen gestartet. Sie empfängt vom Network Testcase Engine Service Tasks, welche ausgeführt werden. Die Ergebnisse werden dem Service geschickt und von da auf die Datenbank geschrieben.
Der Start eines Testcases wird mit einem UDP Multicast Paket auf der Adresse 224.4.4.4 und Port 4444 getriggert.

1.5 ReportingService

Der ReportingService nimmt Nachrichten von verschiedenen Komponenten entgegen und speichert diese in der Datenbank. Diese Daten werden im Webinterface angezeigt.

Abbildung 17: Interface IReport

1.6 Network Testcase Engine Service

Der Network Testcase Engine Service ersetzt das Control Center aus der Semesterarbeit Captive Portal Load Generator. Der Service stellt die Schnittstelle zur Steuerung der Nodes dem Webinterface zur Verfügung.

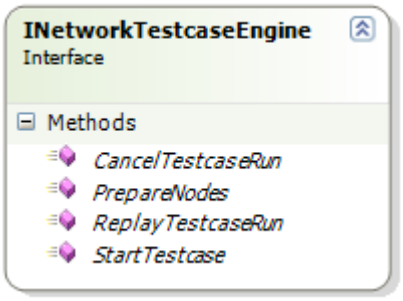


Abbildung 18: Interface INetworkTestcaseEngine

1.7 Webinterface

Über das Webinterface wird die komplette Anwendung gesteuert. Eingesetzt wird das ASP.NET MVC 2 Framework. Als OR-Mapper wird das neue Entity Framework 4.0 eingesetzt.

1.8 MPPHelper

Die MPP Helper Anwendung ist nur in der Entwicklungsphase relevant. Sie wird auf den VMs verwendet, da dort kein Browser zur Verfügung steht. Mit der Anwendung kann man sich beim MPP über die Kommandozeile anmelden.

2. Logical View

Da das Projekt mehrere Komponenten beinhaltet, werden diese auf Namespaces verteilt. Dabei dürfen keine zirkulären Abhängigkeiten entstehen, da die Wartbarkeit sonst drastisch verschlechtert wird.

Namespace	Inhalt	.NET Version
NTE.Service	Service, ersetzt das Control-Center	4.0
NTE.VMRemoteControl	VMRemoteControl Applikation	3.5
NTE.Reporting	ReportingService	4.0
NTE.Model	Generierte Klassen des OR-Mappers	4.0
NTE.Webinterface	ASP.NET MVC2 Webseite	4.0
NTE.AutoUpdater	AutoUpdater Applikation	3.5
NTE.LoadGenerator	LoadGenerator Applikation	3.5
NTE.Virtualisation	DLLs, zBsp. KVMController	3.5
NTE.MPPHelper	MPPHelper Applikation	3.5
NTE.Task	Tasks	3.5

Tabelle 6: Namespaces

Assemblies, welche auch unter Linux verwendet werden, dürfen höchstens mit Version 3.5 des .NET Frameworkes kompiliert werden. Dies Aufgrund der Tatsache, dass Mono nur das .NET 2.0 Framework vollständig unterstützt.

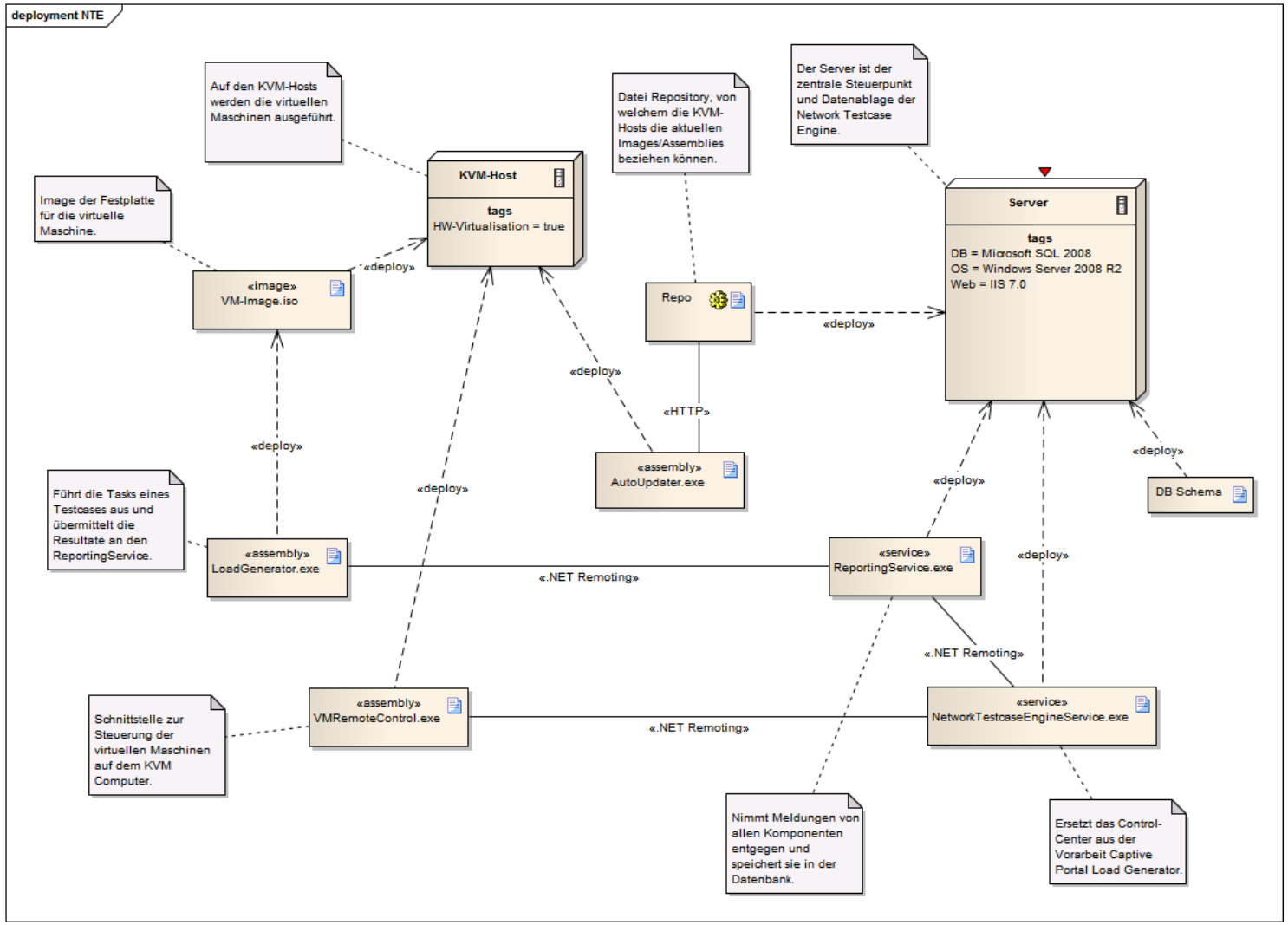
2.1 Überprüfung der Architektur

Das folgende Layer-Diagramm zeigt alle Namespaces mit deren Abhängigkeiten zu anderen Namespaces. Wie bereits erwähnt, dürfen hier keine zirkulären Abhängigkeiten vorhanden sein. Im Visual Studio kann der vorhandene Code mit dem Layer-Diagramm validiert werden. Diese Validation kann der Entwickler von Hand oder automatisch beim Kompilieren auslösen lassen. Beim Einchecken des Sourcecodes wird die Validation erzwungen. Code, welcher dieArchitektur-Richtlinien nicht einhält, wird beim Check-In zurückgewiesen und muss korrigiert werden.

2.2 Layer Diagram

3. Deployment

3.1 Diagramm



3.2 Beispiel Hardwareaufstellung

Folgendes Diagramm zeigt ein Beispiel einer möglichen Hardwareaufstellung. Dem System können beliebig viele KVM Hosts hinzugefügt werden, um die Anzahl Clients zu erhöhen. Für die Durchführung während der Bachelorarbeit, wurden die 3 Server Data Storage, Webserver und Network Testcase Engine Server auf einen physikalische Server gelegt. Beim Einsatz von mehr KVM Hosts, kann es aus Performancegründen sinnvoll sein, diese 3 Server einzel zu betreiben.

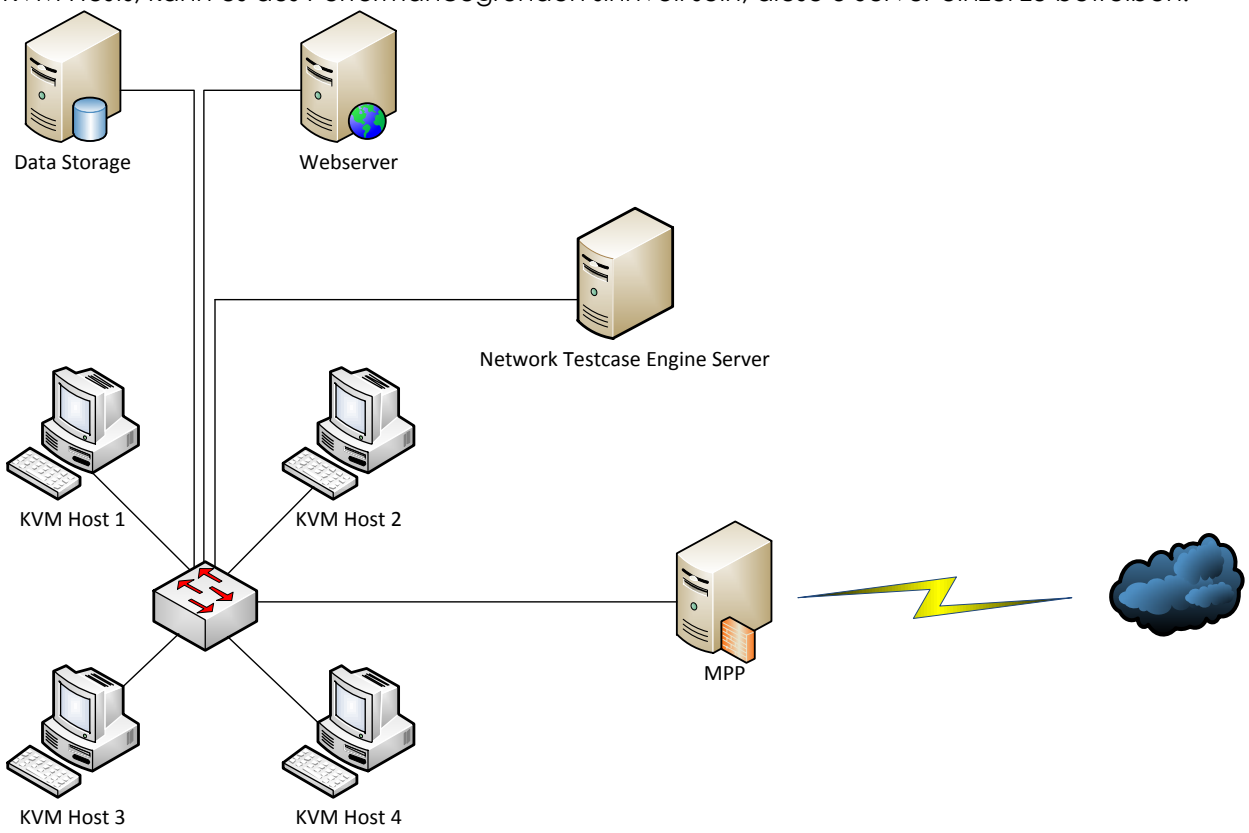


Abbildung 19: Mögliche Hardwareaufstellung

3.3 Automation – AutoUpdater

Bei Änderungen am Code, muss in der Regel nach dem Kompilieren das Resultat auf verschiedene Computer übertragen werden. Aber auch Änderungen am Image für die VMs, müssen auf allen Nodes synchronisiert werden. Dies ist ein sich oft wiederholender Prozess, bei welchem der Mensch zu Fehler neigt und langsamer ist, als ein automatisiertes Skript.
Diese Aufgabe übernimmt deswegen das Programm AutoUpdater. In einem Zentralen Repository (Webserver im Diagramm) werden Dateien und deren MD5 Hashes über HTTP öffentlich gemacht. Zur AutoUpdater Anwendung gehört eine Konfigurationsdatei (config.xml), in welcher konfiguriert werden kann, welche Dateien auf Aktualisierungen geprüft werden sollen. Ob im Repository eine neuere Version der konfigurierten Dateien verfügbar ist, ermittelt die Anwendung mit einem Checksummen vergleich.
Um die md5.xml Datei mit den aktuellen Hashes zu generieren, dient die Anwendung ChecksumGenerator. Sie ermittelt zu allen Dateien im aktuellen Verzeichnis die Checksumme und speichert dies im selben Ordner als md5.xml ab.

3.4 Config.xml (Beispiel)

```
<?xml version="1.0" encoding="utf-8"?>
<AutoUpdaterSettings xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <Md5File>md5.txt</Md5File>
  <Repository>http://10.240.0.2:8888/</Repository>
  <Files>
    <File>example_file.txt</File>
  </Files>
</AutoUpdaterSettings>
```

3.5 Md5.xml (Beispiel)

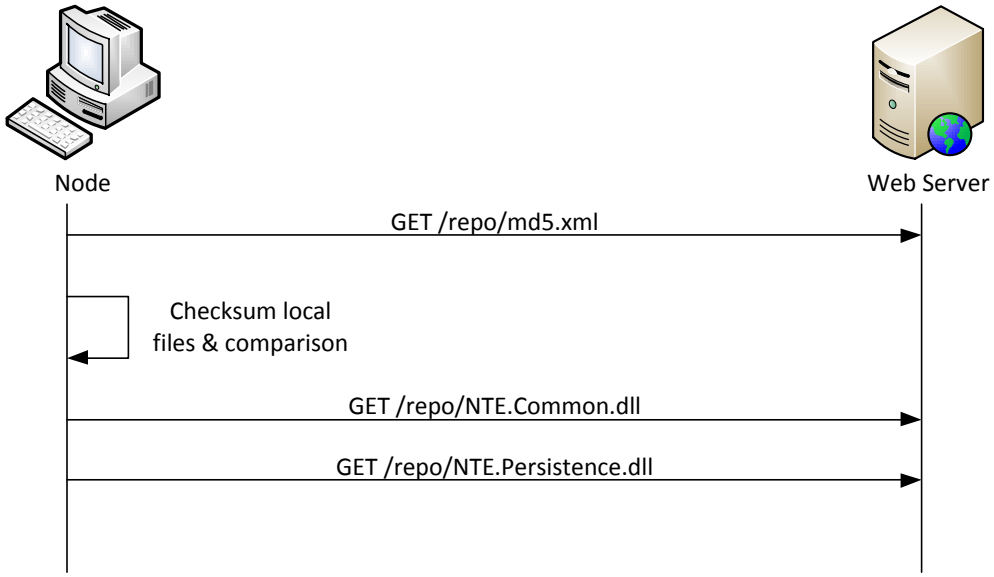
```
<?xml version="1.0" encoding="utf-8"?>
<Md5 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <FileChecksum>
    <Name>File1.txt</Name>
    <Md5>d0089718b62f6e9d91154acae007699c</Md5>
  </FileChecksum>
</Md5>
```

3.6 Vorteile

Durch den Einsatz der AutoUpdater Anwendung und dem zentralen Repository, können beliebige Dateien, unabhängig vom Typ, schnell und automatisiert aktualisiert werden. Die XML Dateien sind sehr einfach und schnell von Hand den Wünschen entsprechend geschrieben. Wenn der AutoUpdater keine Konfigurationsdatei findet, generiert es die oben abgebildete config.xml Datei als Template automatisch.

3.7 Beispielseinsatz

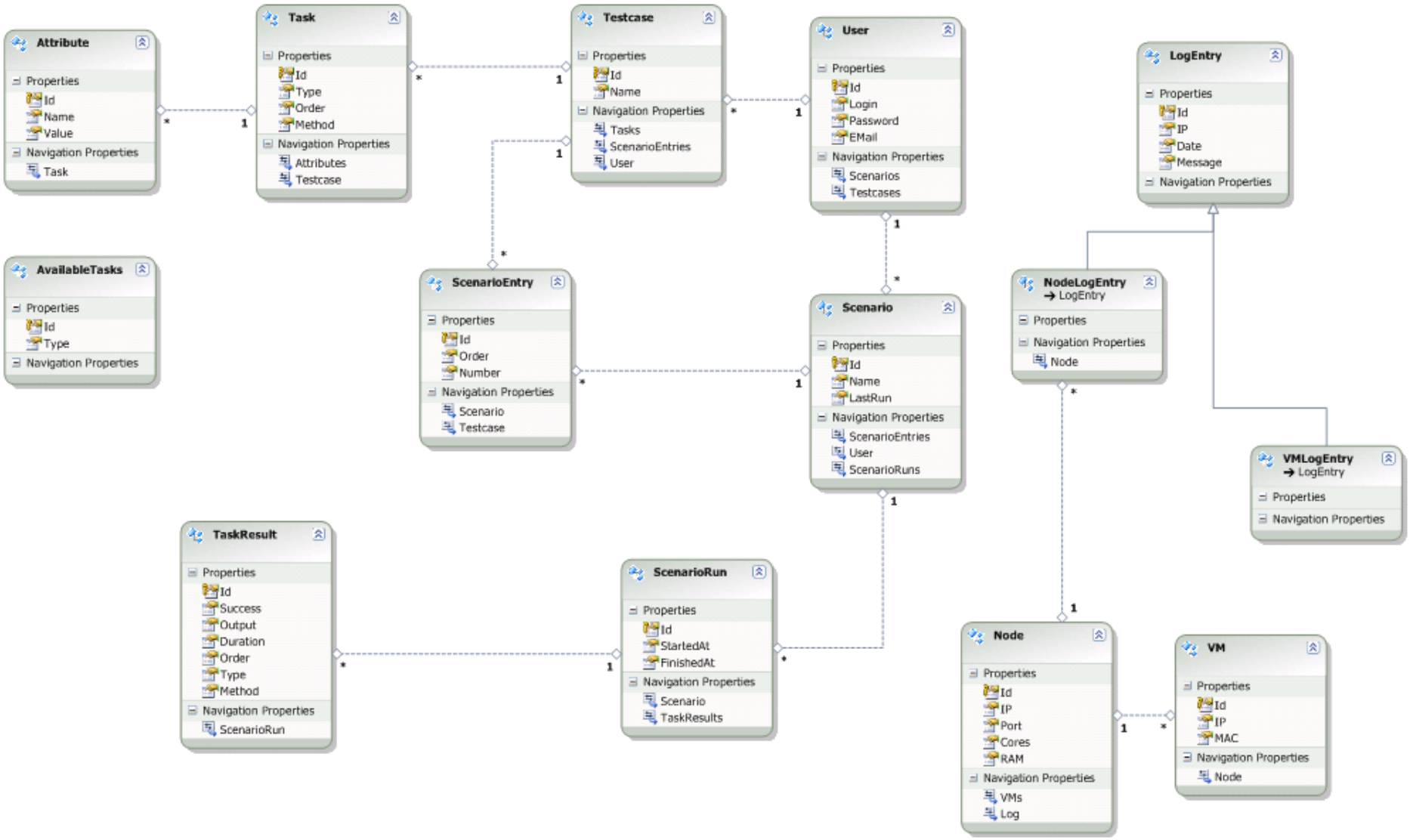
Folgendes Sequenzdiagramm zeigt einen Beispielsablauf, wenn der AutoUpdater auf einer Node ausgeführt wird:



4. Persistence View

Als Persistenzframework wird das Entity Framework von Microsoft eingesetzt. Es ist zur Zeit im RC Stadium und wird mit Visual Studio 2010 und dem .NET Framework 4.0, welche beide auch RC sind, im April 2010 released.
Das Entity Framework ist ein vollwertiger OR Mapper, welcher für verschiedene Datenbankserver eingesetzt werden kann. Eines der wichtigsten Neuerungen gegenüber den bisherigen auf ADO.NET aufbauenden Frameworks, ist die Möglichkeit, mit einem leeren Schema zu beginnen und durch das Schema die Datenbank zu generieren. Bisher war es nur möglich, das Schema von der Struktur der Datenbank zu generieren.
Die Datenbank wird im Designer von Visual Studio erstellt. Das Diagramm wird in DDL umgewandelt, damit kann die Datenbank auf dem Server erstellt werden.

4.1 Diagramm


Attribute
Properties
Id
Name
Value
Navigation Properties
Task
Task
Properties
Id
Type
Order
Method
Navigation Properties
Attributes
Testcase
Testcase
Properties
Id
Name
Navigation Properties
Tasks
ScenarioEntries
User
User
Properties
Id
Login
Password
EMail
Navigation Properties
Scenarios
Testcases
LogEntry
Properties
Id
IP
Date
Message
Navigation Properties
AvailableTasks
Properties
Id
Type
Navigation Properties
ScenarioEntry
Properties
Id
Order
Number
Navigation Properties
Scenario
Testcase
Scenario
Properties
Id
Name
LastRun
Navigation Properties
ScenarioEntries
User
ScenarioRuns
NodeLogEntry
LogEntry
Properties
Navigation Properties
Node
VMLogEntry
LogEntry
Properties
Navigation Properties
TaskResult
Properties
Id
Success
Output
Duration
Order
Type
Method
Navigation Properties
ScenarioRun
ScenarioRun
Properties
Id
StartedAt
FinishedAt
Navigation Properties
Scenario
TaskResults
Node
Properties
Id
IP
Port
Cores
RAM
Navigation Properties
VMs
Log
VM
Properties
Id
IP
MAC
Navigation Properties
Node

5. Kommunikation & Abläufe

In diesem Kapitel werden die verwendeten UDP Adressen und Ports, sowie die Ports der .NET Remoting Services definiert. Weiter werden feste Abläufe spezifiziert, welche die Komponenten kennen müssen, um miteinander zu kommunizieren.

5.1 Abläufe

5.1.1 Node Discovery

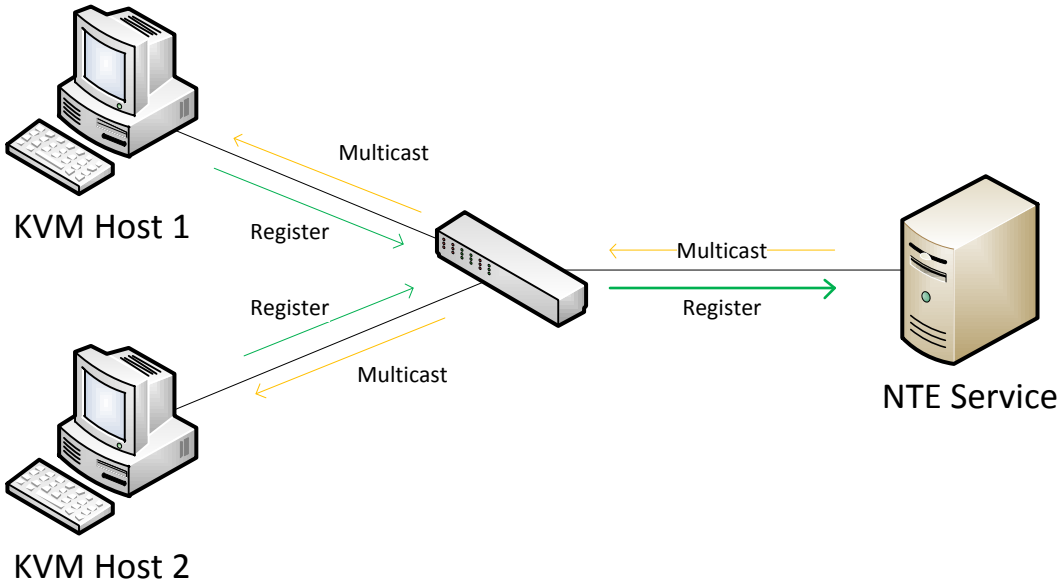


Abbildung 20: Node Discovery mit Multicast

Um den Konfigurationsaufwand möglichst klein zu halten, sollen alle Nodes im System automatisch gefunden werden können.
Der Network Testcase Engine Service schickt auf der Multicast Adresse 224.4.4.4, Port 4444, den ASCII String „request_register“. Die VMRemoteControl Anwendung reagiert auf dieses Multicastpaket, indem es beim Sender die Remote-Methode `RegisterNode(string ip, int port, Int64 ram, int cores)` aufruft. Als Parameter werden Hardwaredetails übergeben, damit später die VMs optimal verteilt und gestartet werden können.

Parameter	Beschreibung
IP	IP Adresse der Node. Wird vom Service später für Remote-Aufrufe verwendet.
Port	Port, auf welchem der IVMControl Service erreichbar ist. Standard ist 4444.
RAM	Verfügbarer Arbeitsspeicher der Node in MB. Wird vom Service zur optimalen Aufteilung der VMs verwendet.
Cores	Die Anzahl logischer Cores der Node. Wird vom Service für das Aufstarten der VMs verwendet. Je mehr Cores eine Nodes hat, desto mehr VMs werden gleichzeitig vom Service auf der Node gestartet.

Tabelle 7: Parameterbeschreibung RegisterNode()

5.1.2 Installation und Verteilung der VMs auf die Nodes

Die Installation einer virtuellen Maschine dauert auf einer Node weniger als eine Sekunde (Es muss lediglich ein Overlay-Image vom Base-Image erstellt werden).
Beim Start eines Szenarios verbindet sich der NTE Service mit allen VMRemoteControl Anwendungen auf den Nodes und erstellt alle virtuellen Maschinen. Die VMs werden gleichmässig auf alle Nodes aufgeteilt, um die Wartezeit, bis alle VMs gestartet sind, möglichst kurz zu halten. Selbst bei mehr als 20 Nodes und über 1000 VMs benötigt die Installation lediglich ein paar Sekunden.

5.1.3 Starten der VMs

Die VMs sind so konfiguriert, dass die LoadGenerator Anwendung automatisch nach dem Bootprozess gestartet wird. Diese meldet sich als erstes beim NTE Service, dass er bereit ist. Als Antwort erhält die Anwendung vom Service eine Liste von Tasks, welche der LoadGenerator abarbeiten soll.

5.2 Multicast Adressen / Port Tabelle

Adresse / Port	Beschreibung
224.4.4.4:4444	An diese Multicast-Adresse sendet der NTE Service den ASCII String „request_register“. Die Nodes reagieren darauf und registrieren sich beim NTE Service.
224.5.5.5:5555	An diese Multicast-Adresse wird der ASCII String „go“ gesendet, um den LoadGenerators den Startbefehl zu geben.
TCP 4444	Standardport für alle .NET Remoting Schnittstellen (IVMControl, IReport und IRegister).

6. Task API

Durch die Task API können neue Tasks für die Network Testcase Engine entwickelt werden, ohne dass Kenntnisse der Architektur nötig sind. Beim Architekturentscheid wurde viel Wert auf eine hohe Kohäsion gelegt. So muss nun eine Task-Implementation sich nur um die Logik des Tasks kümmern. Alles weitere übernimmt das Webinterface und die LoadGenerator Anwendung. Damit dies möglich ist, müssen ein paar Richtlinien eingehalten werden.

6.1 Allgemein

Um ein Task zu implementieren, muss die Assembly NTE.Common.dll referenziert werden. Sie enthält die Basisklasse aller Tasks und die benötigten Attribute.
Ein Task muss unter Linux mit Mono lauffähig sein. Somit muss die Assembly mit der .NET 2.0 Runtime kompatibel sein. Trotzdem können aber im Code die neuen Kompilerfeatures der Version 3.5, wie zum Beispiel Linq, verwendet werden.
Folgende Kriterien müssen erfüllt sein:

- Die Task Klasse muss von NTE.Common.Task.TaskBase ableiten
- Die Task Klasse muss im Namespace NTE.Tasks sein
- Der Code muss als DLL kompiliert werden
- Namenskonvention der DLL: NTE.Tasks.Taskname.dll

6.2 Methoden

Jede Methode, welche in einem Testcase als Task ausgeführt werden können soll, muss mit dem Attribut TaskMethod attributiert werden. Task Methoden müssen als Rückgabetyp bool haben, um dem LoadGenerator mitzuteilen, ob die Ausführung fehlerfrei war.
Damit das Webinterface weiss, welche Methode welche Properties verwendet, werden die Methoden zusätzlich mit dem Attribut RequiredProperty attributiert.

6.3 Properties

Propreties, welche im Webinterface konfiguriert werden sollen, müssen mit dem Attribut TaskProperty attributiert werden.

6.4 Beispiel

Folgendes Beispiel zeigt die Ping Task Klasse mit allen notwendigen Attributierungen:

```
namespace NTE.Task.Ping
{
    public class Ping : TaskBase
    {
        public Ping()
        {
        }

        [TaskProperty()]
        public string IP { get; set; }

        [TaskProperty()]
        public int Times { get; set; }

        [TaskMethod()]
        [RequiredProperty("IP")]
        [RequiredProperty("Times")]
        public bool PingIP()
        {
            return true;
        }
    }
}
```

7. Virtualisierung

In diesem Kapitel wird auf die verwendete Virtualisierungssoftware Kernel-based Virtual Machine (KVM) eingegangen. Die optimale Lösung (KVM, VMWare, Xen oder UML) wurde bereits in der vorangehenden Semesterarbeit evaluiert. In dieser Bachelorarbeit bauen wir auf dem Ergebnis der SA auf und verwenden deshalb KVM.

7.1 Kernel-based Virtual Machine

Kernel-based Virtual Machine, kurz KVM wurde im Oktober 2006 veröffentlicht und ist ab Version 2.6.20 des Linux-Kernels als Modul enthalten. Entwickelt wurde es von dem israelischen Unternehmen Qumranet, welches im September 2008 von Red Hat aufgekauft wurde.
KVM selbst nimmt keine Emulation vor, sondern stellt nur die Infrastruktur dazu bereit. Ein modifiziertes QEMU ist zur Zeit die einzige Möglichkeit, diese zu nutzen. Nach dem Laden des Modules arbeitet der Linux Kernel selbst als Hypervisor für virtuelle Maschinen.
Mit KVM können von einem bereits bestehenden Image in sehr kurzer Zeit Overlay Images erstellt werden. Auf das Base-Image wird nur lesend zugegriffen und im Overlay Image die Differenzen gespeichert. Änderungen im Overlay-Image können bei Bedarf in das Base-Image comitted werden.

7.2 Wichtige Befehle

7.2.1 Image erstellen

```
kvm-img create -f qcow2 base.img 8G
```

Erstellt ein Image im qcow2 Format, welches maximal 8GB gross werden kann.

7.2.2 Overlay-Image erstellen

```
kvm-img create -b base.img -f qcow2 overlay.img
```

Erstellt ein Overlay Image von base.img. Ein Overlay Image speichert nur die Differenz zum Basisimage. Das Overlay Image ist zu beginn lediglich 1 KB gross und ist sehr schnell erstelt.

7.2.3 Änderungen comitten

```
kvm-img commit -f qcow2 overlay.img
```

Die Änderungen vom overlay.img werden in das base.img comitted.

7.2.4 VM starten (Installation)

```
kvm-img -hda base.img -cdrom DebianInstaller.iso -boot d
```

Startet eine VM, als HDA wird das base.img gemounted. Ebenfalls wird als CD-Laufwerk das DebianInstaller.iso gemounted und beim Booten automatisch ausgeführt (-boot d).

7.2.5 VM starten

```
kvm -hda image.iso -net nic,macaddr=<MAC> -net tap,script=kvm-
ifup,downscript=kvm-ifdown -m 51 –nographic
```

Zum Image müssen TAP Scripts sowie eine MAC Adresse mitgegeben werden. Mit dem Parameter `-m` wird das RAM in MB angegeben. Durch Angabe von `–nographic` wird kein SDL Fenster erzeugt.

7.3 TAP Scripts

TAP ist ein virtueller Netzwerk-Kerneltreiber, welcher eine Netzwerkkarte über Software simuliert. Normalerweise befindet sich hinter einem Netwerkgerät, wie zum Beispiel `eth0`, direkt eine entsprechende Hardware in Form einer Netzwerkkarte. Werden Pakete an ein TAP Gerät gesendet, so werden diese an ein Programm im Userspace weitergeleitet und umgekehrt. Damit ein Programm im Userspace auf das TAP Gerät zugreifen kann, hat dies für das TAP Gerät im Ordner `/dev` eine Gerätedatei mit der Bezeichnung `tapN`.

7.3.1 Ifup Script

Beim Start einer virtuellen Maschine wird folgendes Script ausgeführt:

```
#!bin/sh
set -x

switch=br0

if [ -n "$1" ];then
        /usr/bin/sudo /usr/sbin/tunctl –u `whoami` -t $1
        /usr/bin/sudo /sbin/ip link set $1 up
        sleep 0.5s
        /usr/bin/sudo /usr/sbin/brctl addif $switch $1
        exit 0
else
        echo "Error: no interface specified"
        exit 1
fi
```

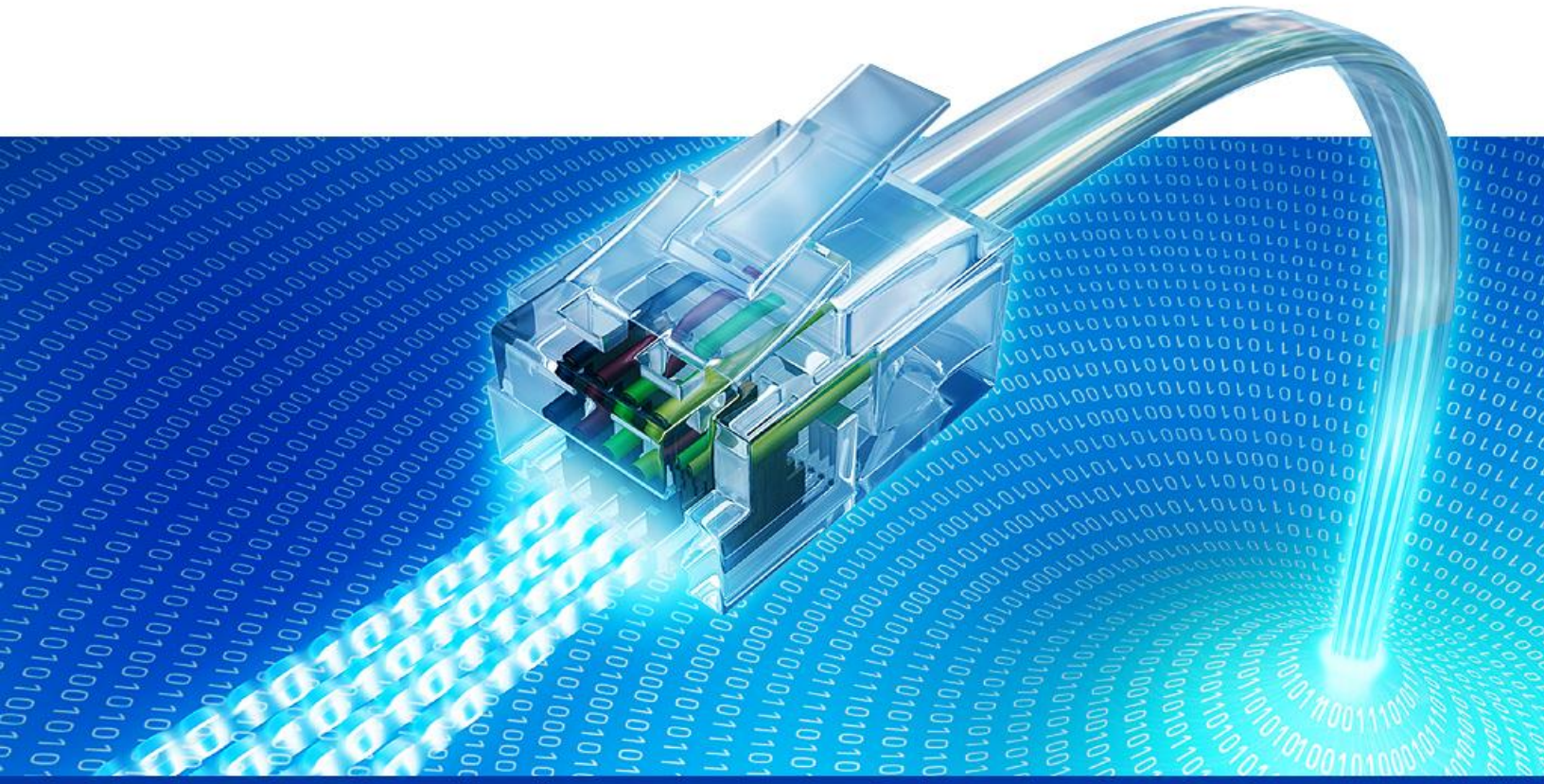
7.3.2 Ifdown Script

Beim herunterfahren einer virtuellen Maschine, wird folgendes Script ausgeführt:

```
#!bin/sh
set -x

switch=br0

if [ -n "$1" ];then
        /usr/bin/sudo /usr/sbin/tunctl -d $1
        sleep 0.5s
        exit 0
else
        exit 1
fi
```



9. Anhang

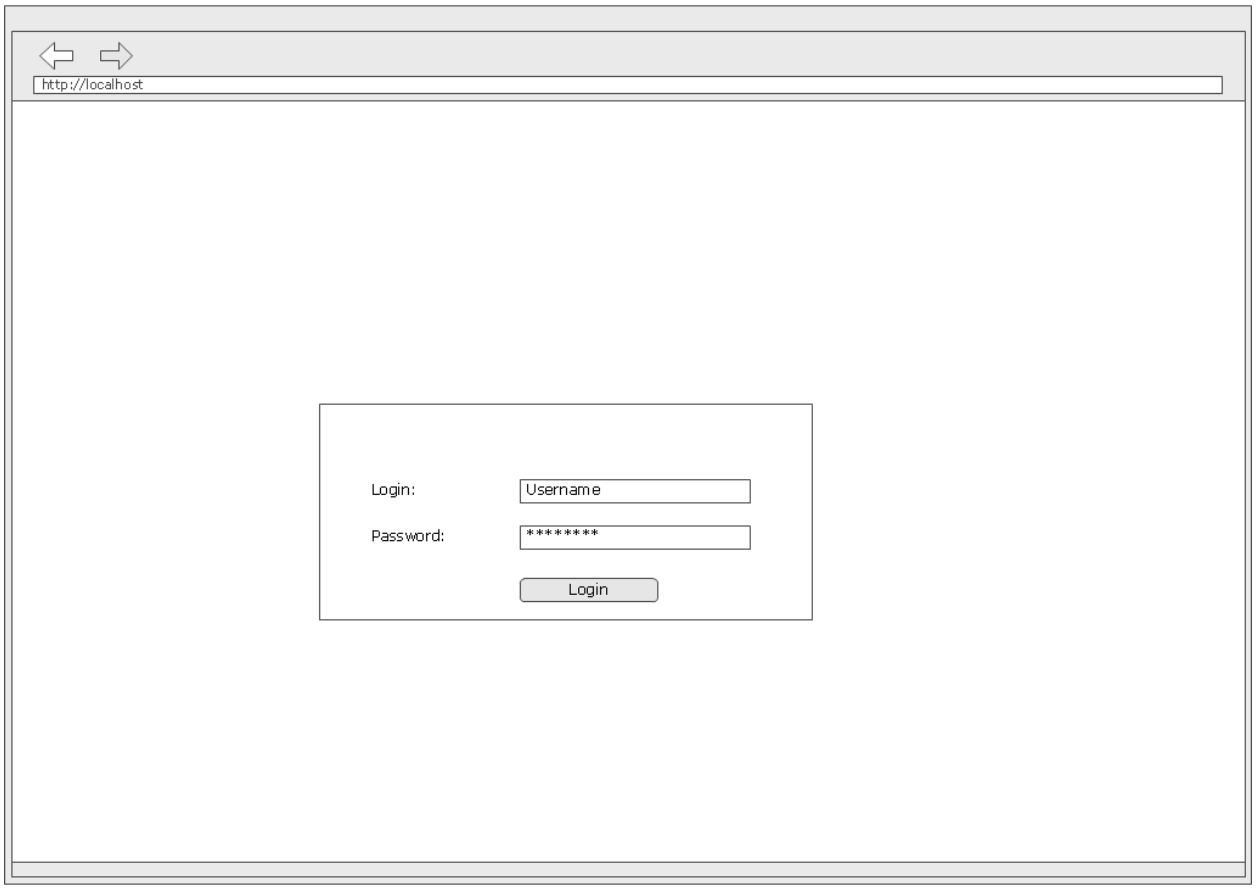
Network Testcase Engine

Bachelorarbeit FS 2010

Technische Hochschule Rapperswil

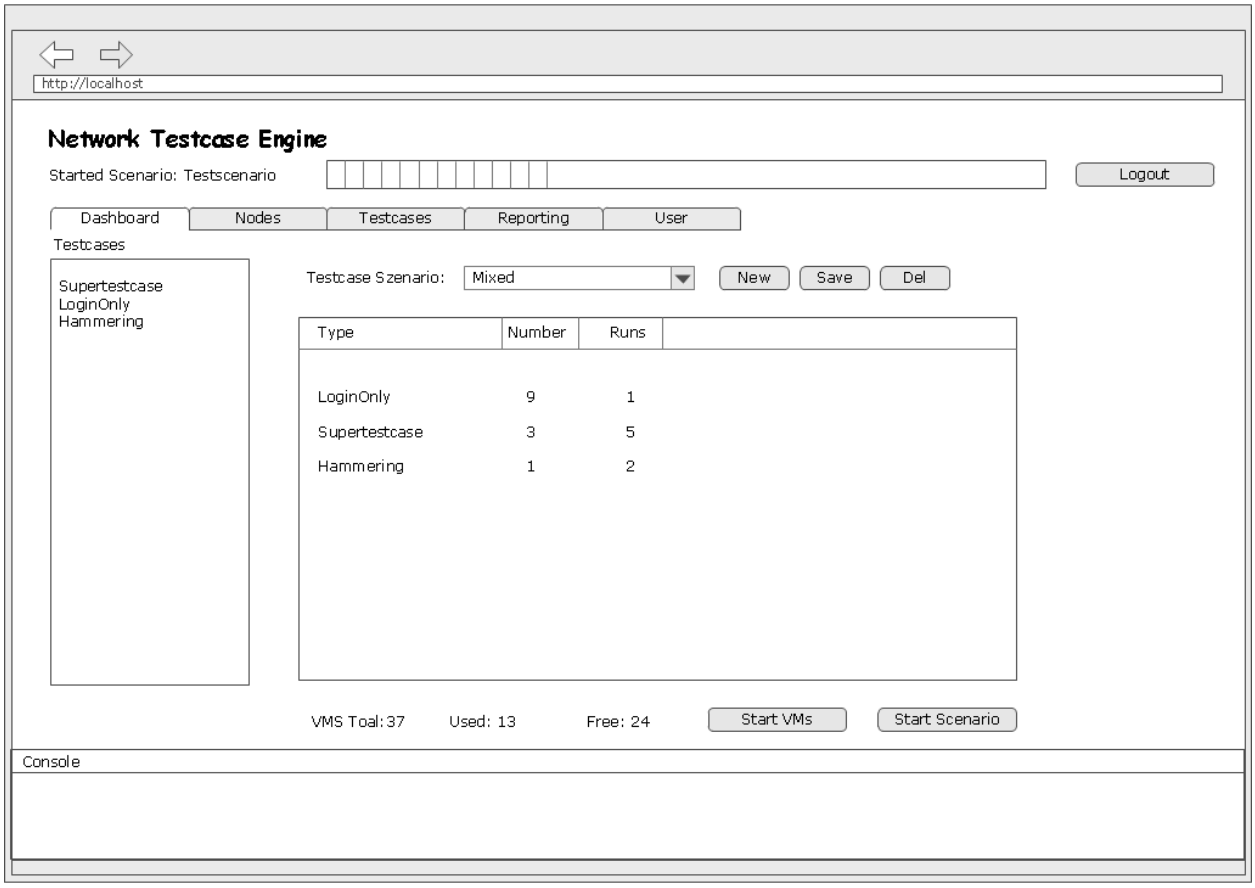
1. Prototyping

1.1 Login



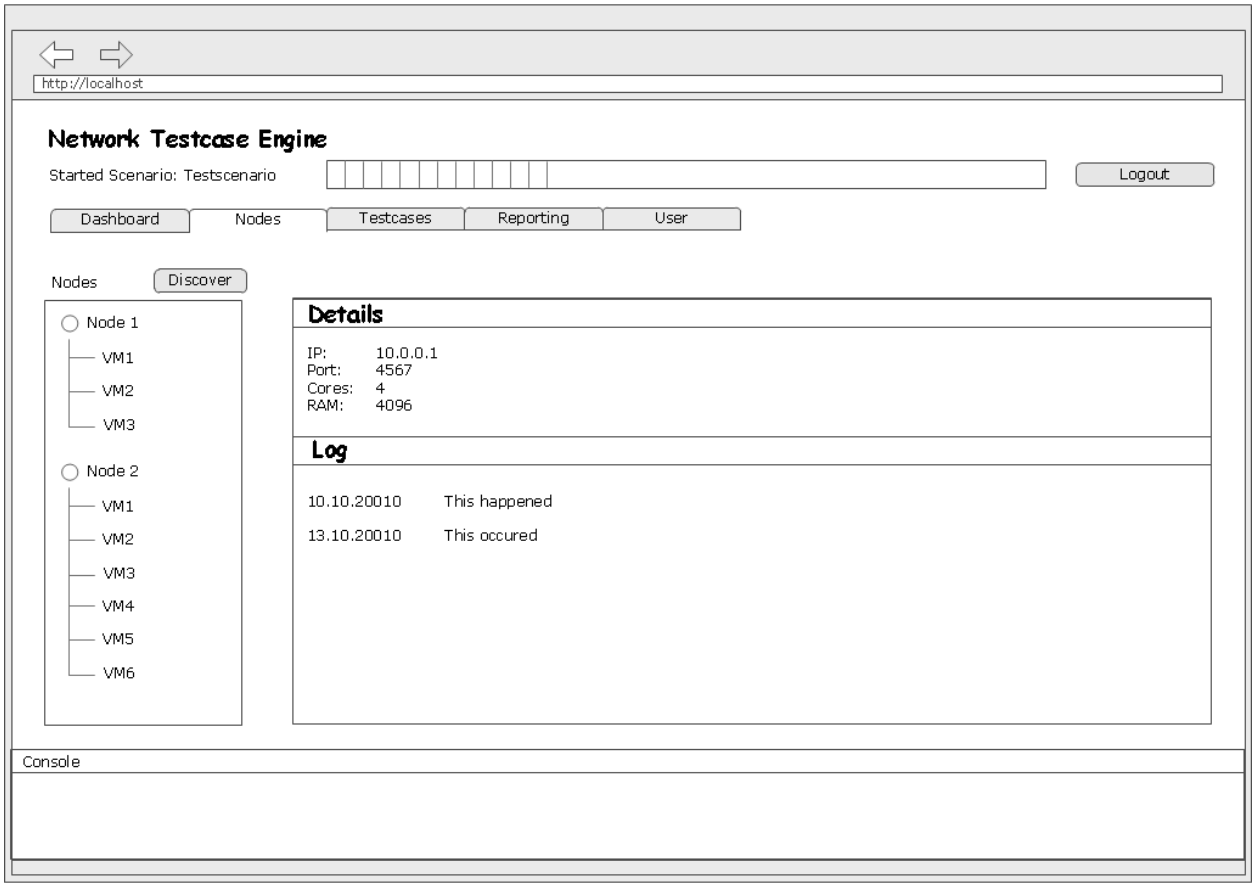
Der Benutzer muss sich einloggen, um auf die Funktionen der Network Testcase Engine zugreiffen zu können.

1.2 Dashboard



Die Verfügbaren Testcases (zuvor im Testcase Editor erstellt) können in ein Szenario eingebettet werden. Dazu gehört die Anzahl an VMs, auf welche der Testcase ausgeführt wird, und wie oft sie wiederholt werden sollen.
Eine Statistik soll anzeigen, wieviele VMs mit den verfügbaren Nodes im Netzwerk gestartet werden können.
Beim starten eines Szenarios werden die VMs automatisch auf die Nodes gleichmässig verteilt.

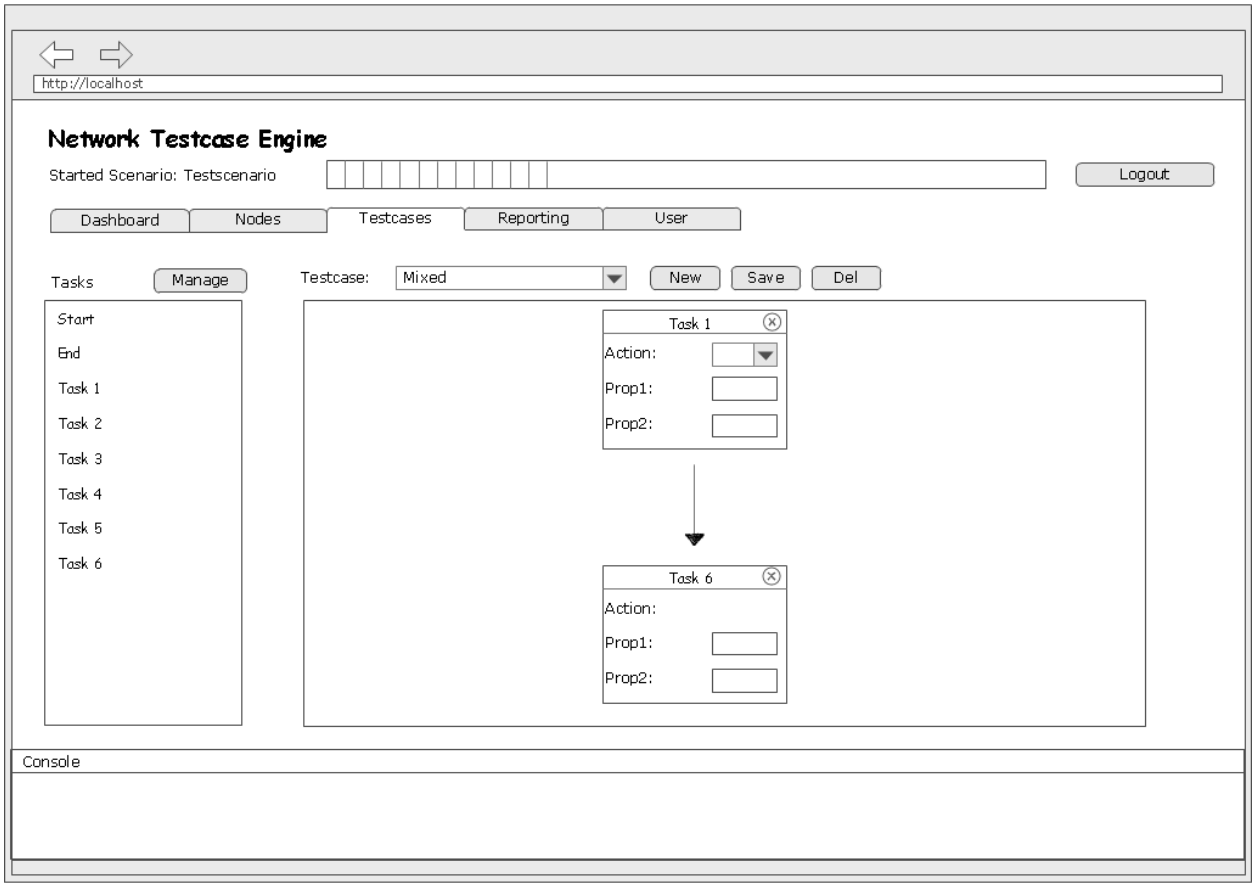
1.3 Nodes



Im Node-Tree kann der Benutzer auf die Logs der einzelnen VMs zugreiffen (durch Klick auf die VM). Zu den Detailinformationen einer VM gehören folgende Punkte:

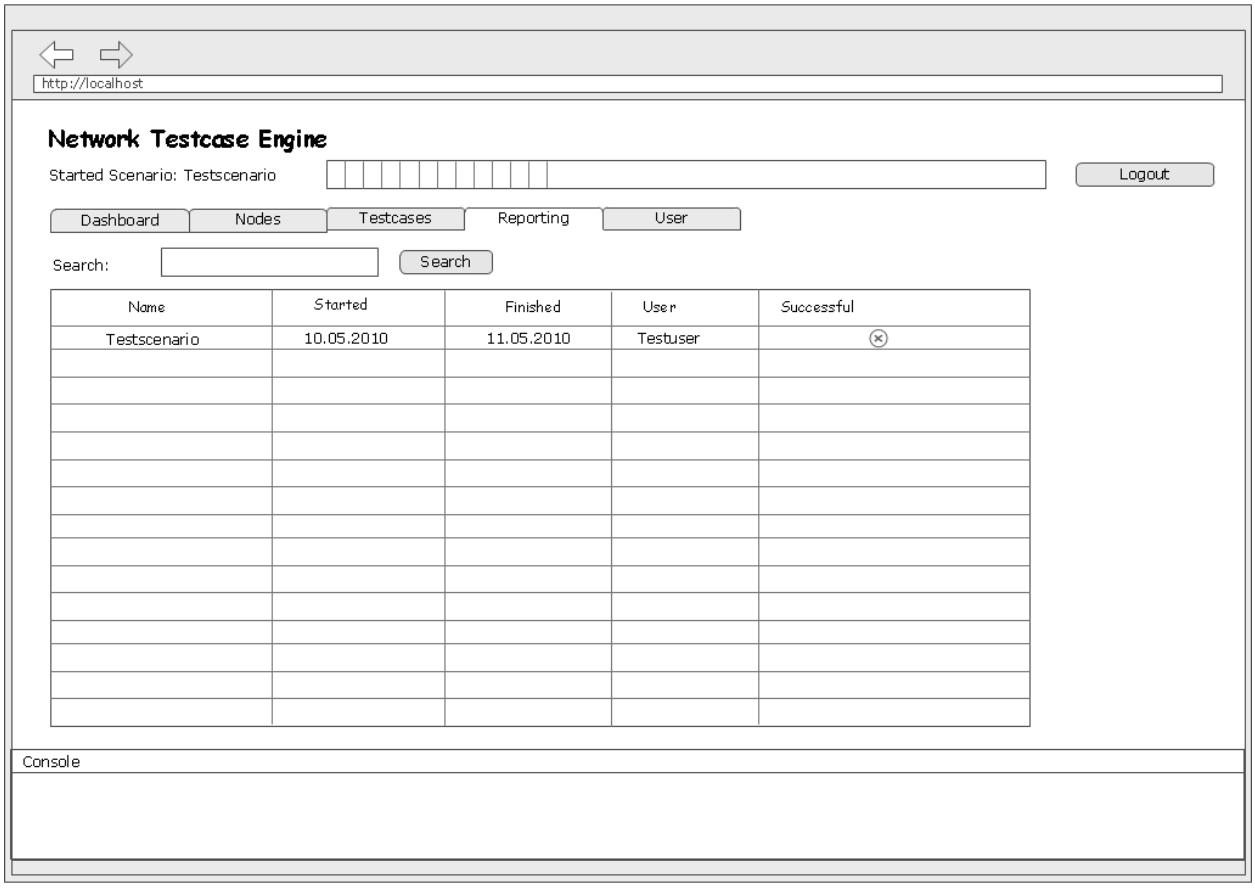
- *IP*
- *Port (für .NET Remoting Zugriff)*
- *Cores*
- *RAM*
- *Maximale Anzahl VMs*
- *Log*

1.4 Testcases



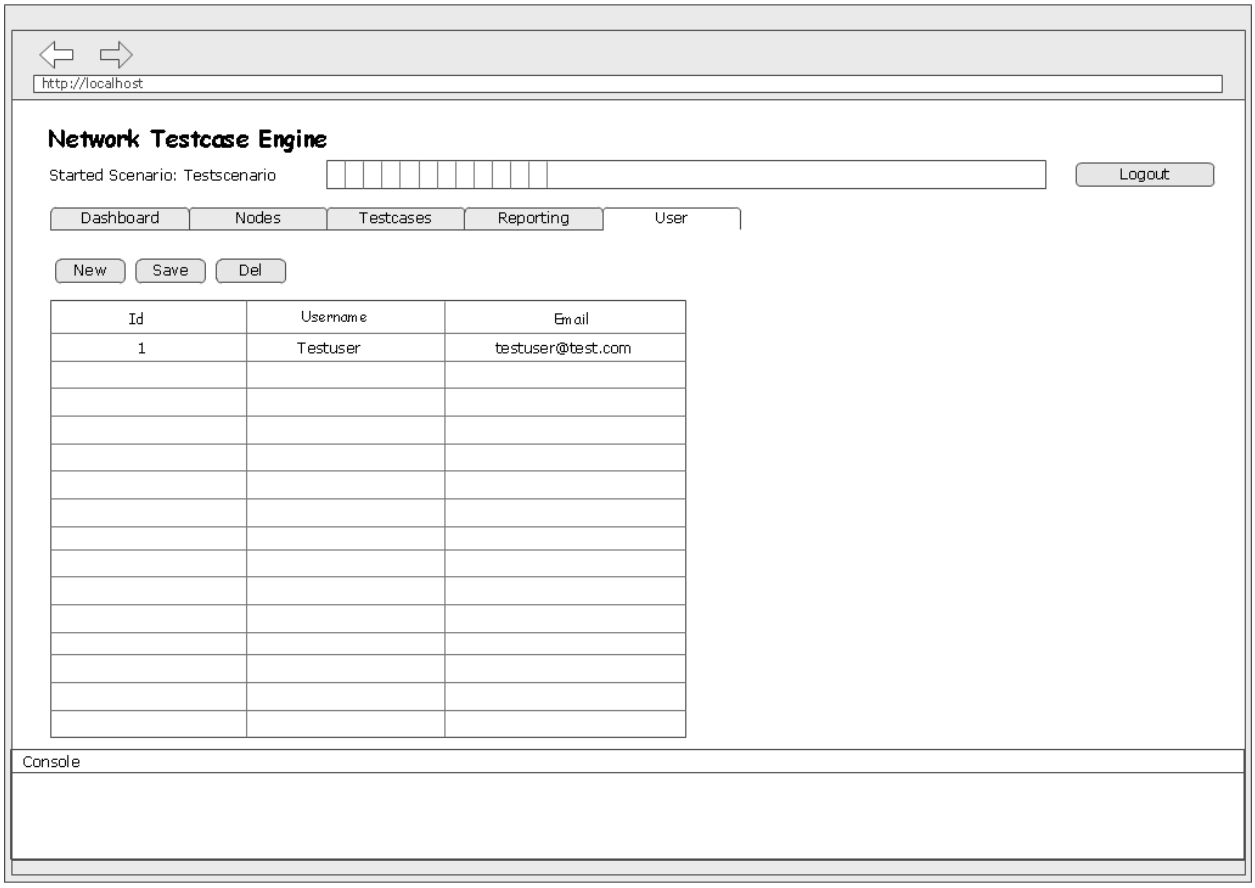
Im Testcase Editor können Testcases erstellt und editiert werden. Die verfügbaren Tasks werden auf der linken Seite aufgelistet und können rechts in das Feld hineingezogen werden. Zu einem Task gehört eine Action und eventuelle Properties, welche gesetzt werden müssen. Die Reihenfolge der Tasks kann per Drag&Drop verändert werden.

1.5 Reporting

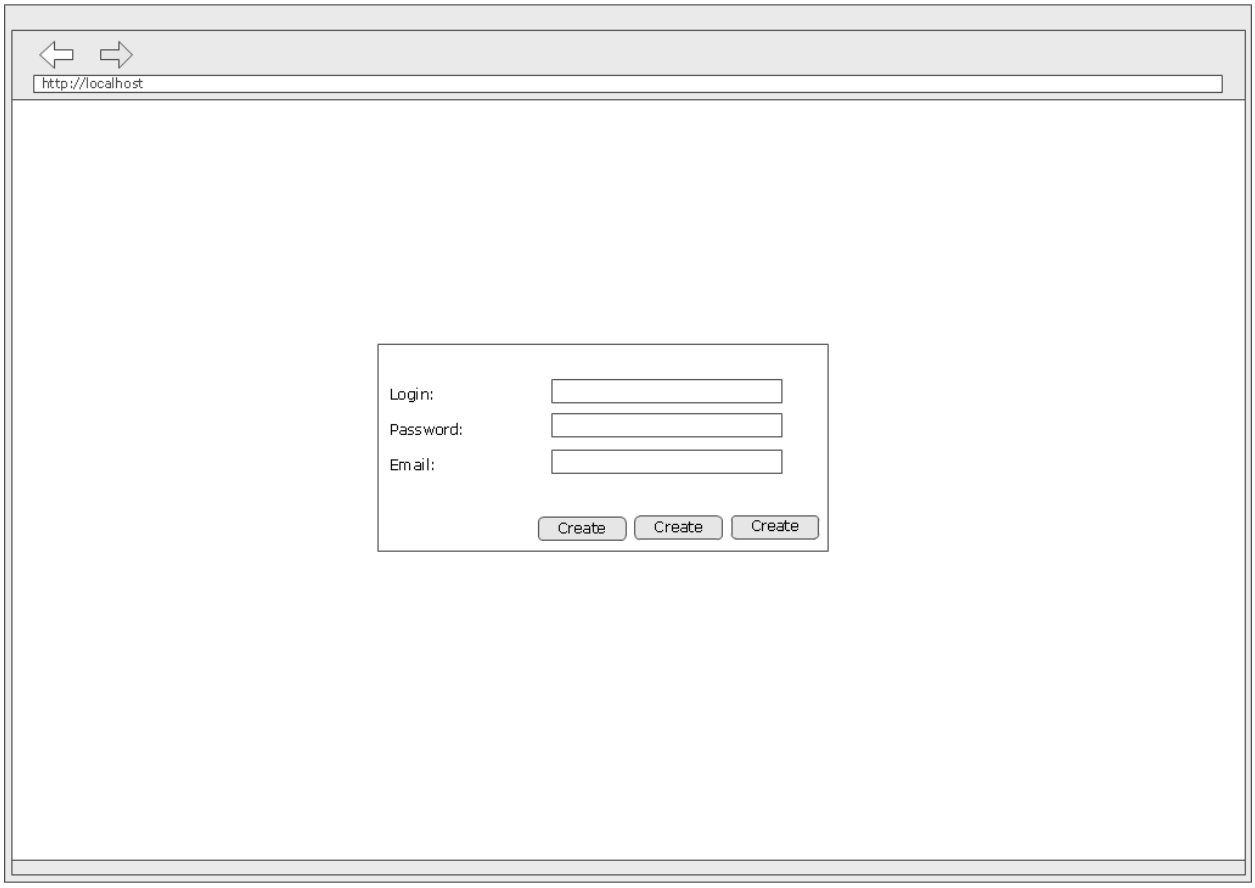


In einer Tabelle sollen alle ausgeführten Tasks eines Szenariodurchlaufes aufgelistet werden. Falls ein Task fehlgeschlagen ist, soll allenfalls eine Fehlernachricht angezeigt werden können.

1.6 Users



Der Zugriff auf die Network Testcase Engine ist nur authorisierten Benutzer gestattet. Hier werden alle Benutzer aufgelistet, und die nötigen CRUD Operationen angeboten.



Formular zum erstellen eines Benutzers.

2. Construction-Phasen

2.1 Construction 1

2.1.1 Bearbeitete User Stories

ID	Titel	Assigned to
15	Benutzer kann Benutzer für das Webinterface verwalten	ymyr
188	LoadGenerator kann Testcase vom NTE Service entgegennehmen und ausführen	oli
189	Reporting Service für allgemeines Logging	oli
191	Benutzer kann Wait Task in Testcase einsetzen	oli
192	Benutzer kann DNSResolver Task in Testcase einsetzen	oli
193	Benutzer kann MPPAction Task in Testcase einsetzen	oli

2.1.2 Bearbeitete Tasks

ID	Titel	Assigned to
187	Szenario Verwaltungsseite erstellen	ymyr
199	Autzentifizierung	ymyr
249	Login	ymyr
250	Logout	ymyr
201	Benutzerverwaltungsseite	ymyr
243	Benutzerübersicht	ymyr
244	Neuen Benutzer erstellen	ymyr
248	JS Validierung Benutzername	ymyr
245	Benutzer editieren	ymyr
246	Benutzer löschen	ymyr
247	Formular für das Kreieren/Editieren eines Benutzers	ymyr
211	Ping Task implementieren	oli
212	Wait Task implementieren	oli
213	DNSResolver Task implementieren	oli
214	MPPAction Task implementieren	oli
235	Basic Logging für Nodes und LoadGenerators	oli
236	Schnittstelle implementieren	oli
239	Basisklasse für Tasks erstellen	oli

241	Webinterface Grundgerüst aufbauen	ymyr
256	OR Mapping mit EF4	oli

2.1.3 Bericht

In der ersten Construction Phase wurden viele Programmieraufgaben erledigt. Da die Entitätsklassen auf Mono laufen müssen und somit nicht mit .NET 4.0 kompiliert werden können, mussten wir vom Design First Ansatz der Datenbank auf den Code First Ansatz umsteigen. Das Objektmodell musste von Hand ohne Designer erstellt und als DLL kompiliert werden. Die Mappings werden in einer weiteren Schicht, welche unter .NET 4.0 läuft realisiert. Durch diese Trennung können nun unsere Assemblies, welche mit Mono unter Linux laufen, die Entities verwenden. Um die Entities zu persistieren, werden sie dem Peristence Layer übergeben, welcher mit der 4.0 Runtime läuft und das Mapping mit dem Entity Framework umsetzt.
Durch die vielen Implementationsaufgaben kam die Dokumentation in dieser Phase etwas zu kurz. Dies wird in der nächsten Phase aufgeholt.

2.1.4 Code Review

Der gesamte Code hat die Überprüfung der Architektur und Coderichtlinien ohne Warnungen bestanden.

2.2 Construction 2

2.2.1 Bearbeitete User Stories

ID	Titel	Assigned to
3	Benutzer kann Test Cases im Webinterface verwalten	ymyr
194	Der Benutzer hat ein Mini-Framework, um weitere Tasks zu erstellen	oli
18	Benutzer kann verfügbare Tasks im Webinterface registrieren	ymyr
190	Benutzer kann Ping Task in Testcase einsetzen	oli
191	Benutzer kann Wait Task in Testcase einsetzen	oli
192	Benutzer kann DNSResolver Task in Testcase einsetzen	oli
193	Benutzer kann MPPAction Task in Testcase einsetzen	oli
251	Benutzer kann Googler Tast in Testcase einsetzen	oli
252	Benutzer kann MPPHammer Task in Testcase einsetzen	oli
202	Benutzer kann Node Discovery durchführen	oli

2.2.2 Bearbeitete Tasks

ID	Titel	Assigned to
197	Attribute von Tasks Dokumentieren	oli
207	Design von Tasks dokumentieren	oli
237	Ausführung von übergebenem Testcase auf VM	oli
240	Informationen an Reporting/Service senden	oli
344	Task via Reflection ausführen	oli
258	Node Discovery implementieren	oli
259	Webseite zur Konfiguration des Durchlaufs erstellen	ymyr

2.2.3 Bericht

In dieser Construction Phase mussten das Defizit der Dokumentation von der letzten Phase aufgeholt werden. Mit dem TeamSpec Word Plugin können nun Work Items aus dem Team Foundation Server in Word verwendet und bei Änderungen abgeglichen werden.
Ziel dieser Construction Phase war es, dass die einzelnen LoadGenerator Tasks ausgeführt werden können. Dies funktioniert einwenig anders, als in der Vorarbeit. Die nötigen Informationen zu den Tasks mit ihren Properties/Methoden werden mit Reflection aus den DLLs herausgeholt (die Attributierung hilft dabei, die entsprechenden Methoden/Properties zu finden).
Ein weiteres Ziel war die Anzeige der Nodes im Webinterface. Die Nodes sollen mit ihren VMs live in einer Tree-View visualisiert werden.

Die Ziele wurden in dieser Construction Phase im Grossen und Ganzen erreicht. Die Tree-View im Webinterface benötigt noch den letzten Feinschliff.
Ende Woche 12 setzen wir einen neuen Meilenstein: Prototyp, mit welchem ein Szenariodurchlauf aus dem Webinterface gestartet werden kann.

2.2.4 Code Review

Der gesamte Code hat die Überprüfung der Architektur und Coderichtlinien ohne Warnungen bestanden.

2.3 Construction 3

2.3.1 Bearbeitete User Stories

ID	Titel	Assigned to
4	Benutzer kann im Webinterface ein Szenario starten	ymyr
202	Benutzer kann Node Discovery durchführen	oli
252	Benutzer kann MPPHammer Task in Testcase einsetzen	oli

2.3.2 Bearbeitete Tasks

ID	Titel	Assigned to
258	Node Discovery implementieren	oli
259	Webseite zur Konfiguration des Durchlaufs erstellen	ymyr
292	Verteilung der VMs auf die Nodes	oli
367	MAC Generator / Verteilung der VMs	oli
254	MPPHammer Task implementieren	oli

2.3.3 Bericht

In dieser Construction Phase wurde viel Zeit in das Webinterface investiert. Das ExtJS benötigt einiges an Einarbeitungszeit, doch die interaktiven Seiten nehmen immer mehr an Gestalt an. Es existieren einige Bugs im ExtJS Framework, welche teilweise schwierig zu finden sind, da sie an Stellen sind, wo man sie nicht erwarten würde (Core Functions). Zum Beispiel liefert getValue() auf einer Combobox oft nichts, obwohl eine Value gesetzt wäre. Der Wert muss selbst über das DOM ausgelesen werden. Die Verteilung der VMs auf die Nodes übernimmt der NTE.Service. MAC Adressen werden automatisch generiert, als Base-Adresse wurde die KVM spezifische DE:AD:BE:EF Adresse genommen.

2.3.4 Code Review

Der Code erfüllt die vorgeschriebenen Architekturrichtlinien. Die Coderichtlinien werden teilweise beim NTE.Model und NTE.Mapping Projekt nicht eingehalten, da mehrere Klassen in einer Codedatei vorhanden sind. Dies muss in der nächsten Construction Phase mit Refactorings behoben werden.

2.4 Construction 4

2.4.1 Bearbeitete User Stories

ID	Titel	Assigned to
3	Benutzer kann Test Cases im Webinterface verwalten	ymyr
4	Benutzer kann im Webinterface ein Szenario starten	ymyr
188	LoadGenerator kann Testcase vom NTE Service entgegennehmen und ausführen	oli
202	Benutzer kann Node Discovery durchführen	oli
18	Benutzer kann verfügbare Tasks im Webinterface registrieren	ymyr

2.4.2 Bearbeitete Tasks

ID	Titel	Assigned to
187	Szenario Verwaltungsseite erstellen	ymyr
346	Task mit Drag&Drop einfügen	ymyr
344	Task via Reflection ausführen	oli

2.4.3 Bericht

Ende dieser Construction Phase sollte eine Demo mit einem ganzen Szenario durchlauf durchgeführt werden. Das Ziel war somit gegeben. Die Entwicklung des Webfrontents geht zwar weiter einwenig harzig vorwärts, dafür ging es im Backend gut vorwärts. Die meisten Funktionen für das Task ausführen konnte in der TaskBase Klasse implementiert werden, um weiterhin eine hohe Kohäsion zu gewährleisten. Die LoadGenerator Anwendung kann nun ein Testcase verarbeiten. Nach dem Multicast go führt es die einzelnen Tasks des Testcases aus und raportiert den Erfolg jedes einzelnen Tasks, damit die Ergebnisse im Webfrontent aufbereitet werden können.
Leider funktionierte die Übergabe eines Testcases mit .NET Remoting nicht. Der Stacktrace beinhaltete die Tiefen des .NET Remoting Frameworks, mit diesen Informationen konnten wir erst nichts anfangen. Der Fehler konnte bis zur Demo nicht gelöst werden, weswegen die Ausführung eines Testcases nicht vorgeführt werden konnte.

2.4.4 Code Review

Der Code erfüllt die vorgeschriebenen Architekturrichtlinien. Der Code, welcher in der letzten Phase Warnungen bei der Überprüfung der Coderichtlinien verursacht hat, wurde mit Refactorings bereinigt.

2.5 Construction 5

2.5.1 Bearbeitete User Stories

ID	Titel	Assigned to
12	Benutzer kann Reports von durchgelaufenen Testcases anschauen.	ymyr

2.5.2 Bearbeitete Tasks

ID	Titel	Assigned to
234	Report Webseite erstellen	ymyr
369	Backend implementieren zum entgegennehmen der Ergebnisse	oli

2.5.3 Bericht

Das Problem bei der Übergabe eines Testcases via .NET Remoting konnte gelöst werden. Ein Referenzierter Datentyp innerhalb eines Testcases war nicht als Serializable markiert, weswegen das Remoting Framework die Serialisierung dieses Types verweigerte. Nachdem der Fehler schnell behoben war, konnte endlich mit dem Testen von Szenariodurchläufen angefangen werden. Auf Windows Systemen liefen die Testcases fehlerfrei und mit dem gewünschten Verhalten durch. Unter Mono gab es immer wieder Probleme. Nach vielen Recherchen kam heraus, dass das Timeout-Handling unter Mono nicht korrekt funktioniert. Nachdem 2 Requests gemacht wurden, werden keine weitere Requests verarbeitet, da der Stack maximal 2 entgegen nimmt. Alle weiteren Requests timen dann automatisch aus. Da Requests teils nicht richtig disposed werden (wie es unter Windows der Fall ist, wenn die Verbindung geschlossen wird), müssen diese explizit disposed werden. So können die Timeout Probleme unter Mono gelöst werden.

2.5.4 Code Review

Der gesamte Code hat die Überprüfung der Architektur und Coderichtlinien ohne Warnungen bestanden.

2.6 Construction 6

2.6.1 Bearbeitete User Stories

ID	Titel	Assigned to
5	Benutzer kann im Webinterface ein durchgeführten Testcase wiederholen	oli
194	Der Benutzer hat ein Mini-Framework, um weitere Tasks zu erstellen	oli
251	Benutzer kann Googler Tast in Testcase einsetzen	oli
253	Googler Task implementieren	oli

2.6.2 Bearbeitete Tasks

ID	Titel	Assigned to
204	Runtime Eigenschaften zwischenspeichern	oli
253	Googler Task implementieren	oli

2.6.3 Bericht

In der letzten Construction Phase sollen noch die letzten Features implementiert werden. Doch bevor die letzten Features implementiert werden, müssen im Frontend einige Verbesserungen gemacht werden. Das Frontent ist nach Anforderung Stateless. Somit muss der NTE.Service den State speichern, da er sonst verloren gehen würde. Die Service-Schnittstelle muss dem Webinterface Methoden zum Abfragen des States zur Verfügung stellen. Änderungen müssen auf der Seite des Services streng synchronisiert werden, da mehrere Clients zur selben Zeit auf dem Frontend sein können.
Es kam zu vielen kleinen Komplikationen, um das Stateless Design umzusetzen. Wir haben zu spät erkannt, dass herkömliche Comet Implementationen Race Conditions aufweisen. Die Zeit wurde zu knapp, um eine richtige Push Implementierung ohne Race Conditions umzusetzen. So blieben wir beim Polling um die Daten aktuell zu halten.
Das Frontend hat zu viel Zeit geraubt, weswegen wir das Replayer-Feature offen lassen müssen.

2.6.4 Code Review

Der gesamte Code hat die Überprüfung der Architektur und Coderichtlinien ohne Warnungen bestanden.

2.7 Fazit

Rückblickend auf die Construction Phasen, würde beim nächsten Mal nicht mehr Javascript zum Einsatz kommen. Javascript bietet zwar viele Möglichkeiten eine Webseite interaktiv zu gestalten, jedoch lebt man gleichzeitig alle Nachteile einer Scripting Sprache. Vielleicht hätten sich gewisse Probleme durch den Einsatz eines anderen (besseren?) Frameworkes vermeiden lassen, doch die Probleme von Skriptingsprachen sind auch dann noch da.
Durch den erhöhten Zeitbedarf bei der Webapplikation, kam das Replayer Feature zu kurz, und musste am Ende offen gelassen werden. Dies ist aber auch aufgrund der späten Erkenntnis, dass normale Comet implementationen Race Conditions aufweisen. Gerne hätten wir eine Comet Lösung ohne Race Conditions implementiert, jedoch blieb am Ende einfach keine Zeit mehr übrig. So waren wir gezwungen, die Daten mit Polling in der Webapplikation aktuell zu halten.
Der Einsatz des .NET Frameworkes hat sich, wie schon bei der Semestararbeit, sehr bewährt. Das .NET Remoting Framework hat uns viel Arbeit bei der Interprozesskommunikation (IPC) abgenommen.
Ebenfalls können wir nur positives Feedback zum Entity Framework 4 CTP3 geben. Die Neuerungen im Entity Framework wie zum Beispiel der Code First Ansatz funktioniert hervorragend. Gleichzeitig war es für uns ein „must-have“ Feature, da die Entitätsklassen in 2 verschiedenen Runtimes gleichzeitig funktionieren mussten.

3. Glossar

A

API	Application Programming Interface => Programmierschnittstelle
ASP.NET	Active Server Pages .NET. Framework für Webanwendungen von Microsoft.
Assembly	Übersetzte Programmklassen als ausführbares Programm oder DLL

C

Client	Programm welches mit einem Server kommuniziert um Dienstleistungen zu nutzen
CLR	Common Language Runtime
Concurrency	Nebenläufigkeit, d.h. Zweit Threads erledigen Arbeiten ohne Abhängigkeiten
COW	Copy On Write

D

DLL	Dynamic Link Library, bezeichnet allgemein eine Dynamische Bibliothek

E

EF	Entity Framewok (ORM)
ExtJS	Javascript Framewort

F

Framework	Programmiergeürst
FS	Frühjahr Semester

G

Google	Bekannte Suchmaschine, www.google.com

H

Host	Beherbergt Clientsoder Server
HSR	Hochschule Rapperswil
HTML	Hypertext Markup Language ist eine Sprache, welche zur Strukturierung von Inhalten wie, Bildern, Texten oder Verweisen in Dokumenten dient.
HTTP Request	Anforderung an eine Web Ressource (HTTP)
HTTP Response	Gibt eine Antwort von einer Internetressource zurück
HW	Hardware

I

Image	Speicherabbild, z.B. einer kompletten Festplatte
INS	Institute for networked solutions
Interface	Schnittstelle, welche gemeinsam e Methoden und Funktionen definiert die in Unterschiedlichen Klassen impementiert werden.

J

Javascript	Skriptsprache, welche hauptsächlich in Webbrowsern eingesetzt wird

K

Kernel	Zentraler Bestandteil eines Betriebssystemes
KVM	Kernel Based Virtual Machine

L

Link	Verweis

M

MPP	Multi Provider Portal
Multicast	Überträgt eine Nachricht von einem Punkt zu einer Gruppe
MS	Meilenstein

N

Namespace	Abstrakter Container welcher logische gruppen , z.B. Klassen, enthält.
NTE	Network Testcase Engine

O

ORM	Object-Relational Mapping. Ermöglicht das ablegen der Objekte eines Programms in eine relationale Datenbank
Overlay-Image	Gepseicherte Differenzen zum Basisimage

P

Prototyping	Methode, welche ein frühzeitges Feedback bezüglich einer Lösung ermöglicht.
Prozess	Ablaufendes Programm

Q

qcow 2	Format für eine Image

R

RTM	Ready To Manufacture

S

Service	Ein Mechanismus um Zugriff auf eine oder mehrere Ressourcen zu haben
Server	Programm welches auf Clients wartet , welche für ihn Dienstleistungen erfüllen
Shadow Copies	Entwickelte Technologie von Microsoft um Snapshots zu erstellen der Daten auch wenn diese gelockt sind.
Surfer	Charakter mit verschiedenen Eigenschaften wie er im Internet Surft
String	Zeichenkette
SVN	Subversion
SW	Software
Szenario	Definiert Surfertypen und wie viele von denen gebraucht warden

T

Team Seite	Jedes Projekt auf dem Team Foundation Server hat ein Portal resp. Team Seite
TFS	Team Foundation Server
Thread	Teil eines Prozesses

U

UML	User Mode Linux
URL/URI	Identifizieren und lokalisieren Ressourcen. Im Falle von URL über das HTTP oder FTP Protokoll

V

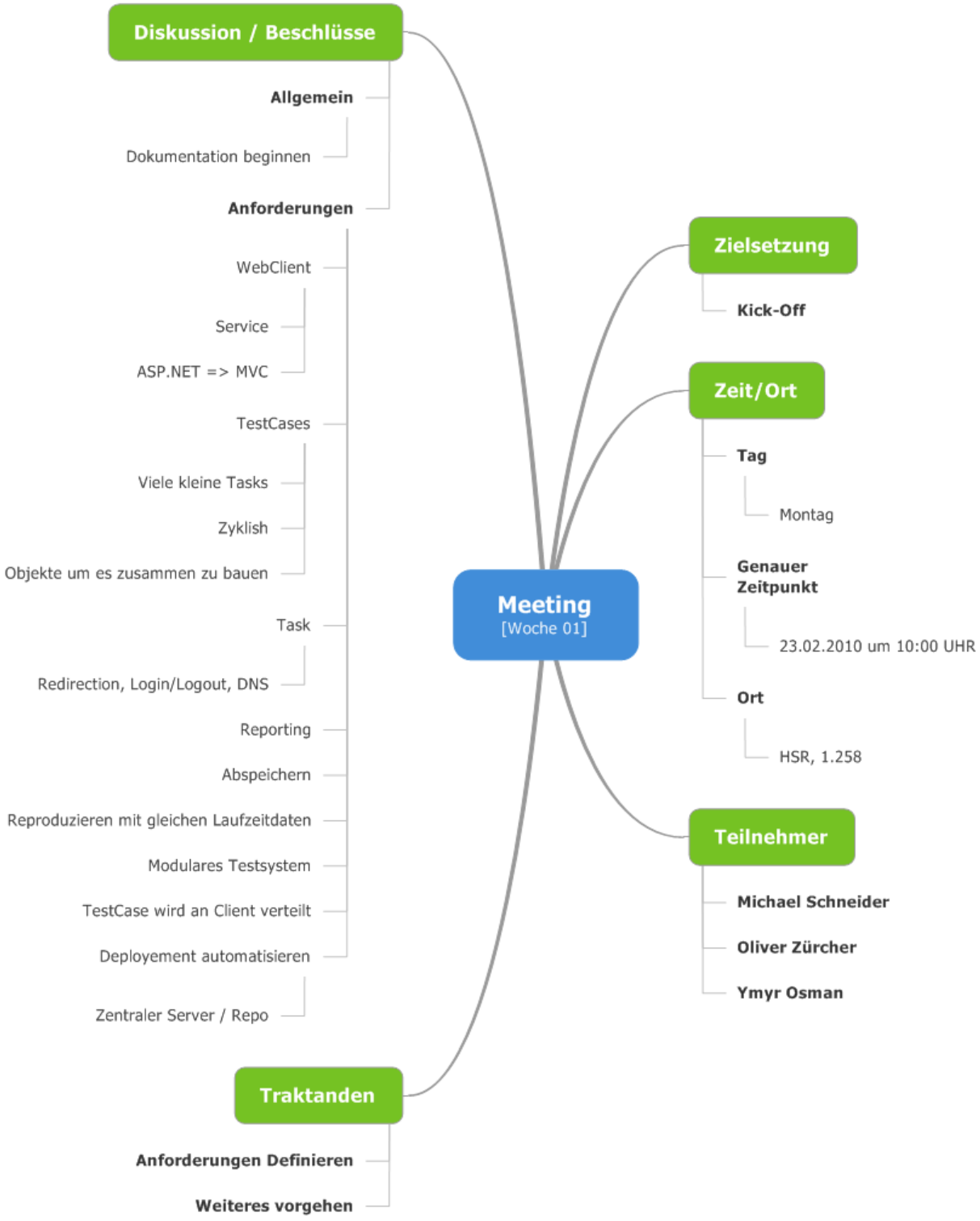
VOIP	Voice over IP
VM	Virtuelle Maschine
VNUML	Virtual Network User Mode Linux

4. Literaturverzeichnis

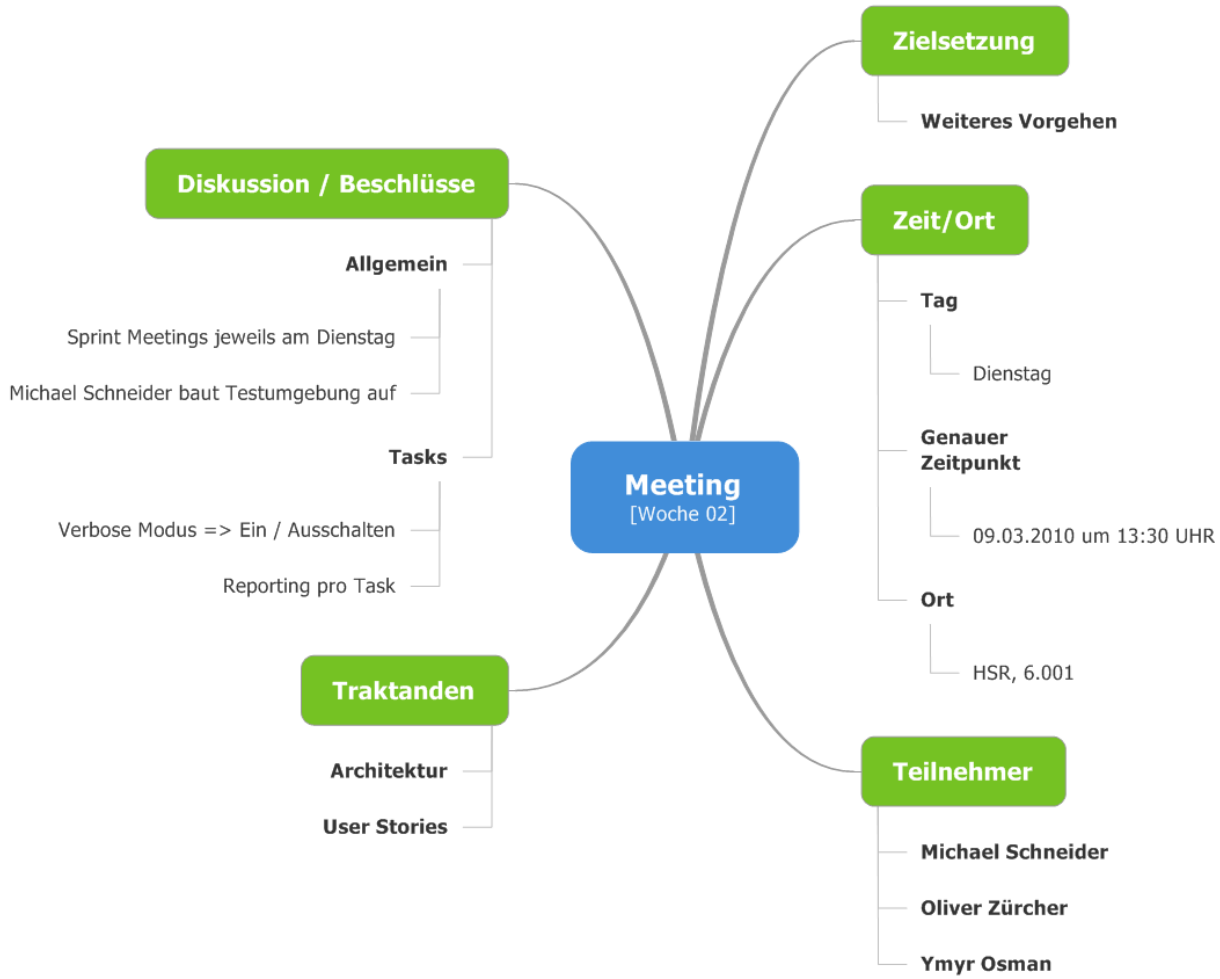
ExtJs API. (2010). Von http://www.sencha.com/deploy/dev/docs/ abgerufen
ExtJs Forum. (2010). Abgerufen am 2010 von http://www.sencha.com/forum/
Kühnel, A. (2008). *Visual C# 2008.* Galileo Computing.
Microsoft. (2010). Microsoft Developer Network.
Obermeyer, P., & Hawkins, J. (16. 02 2002). Microsoft .NET Remoting: Ein technischer Überblick.
Oliver Zürcher, Y. O. (2009). *Semesterarbeit Captive Portal Load Generator*. Von HSR E-Prints: http://eprints.hsr.ch/39/ abgerufen

5. Sitzungsprotokolle

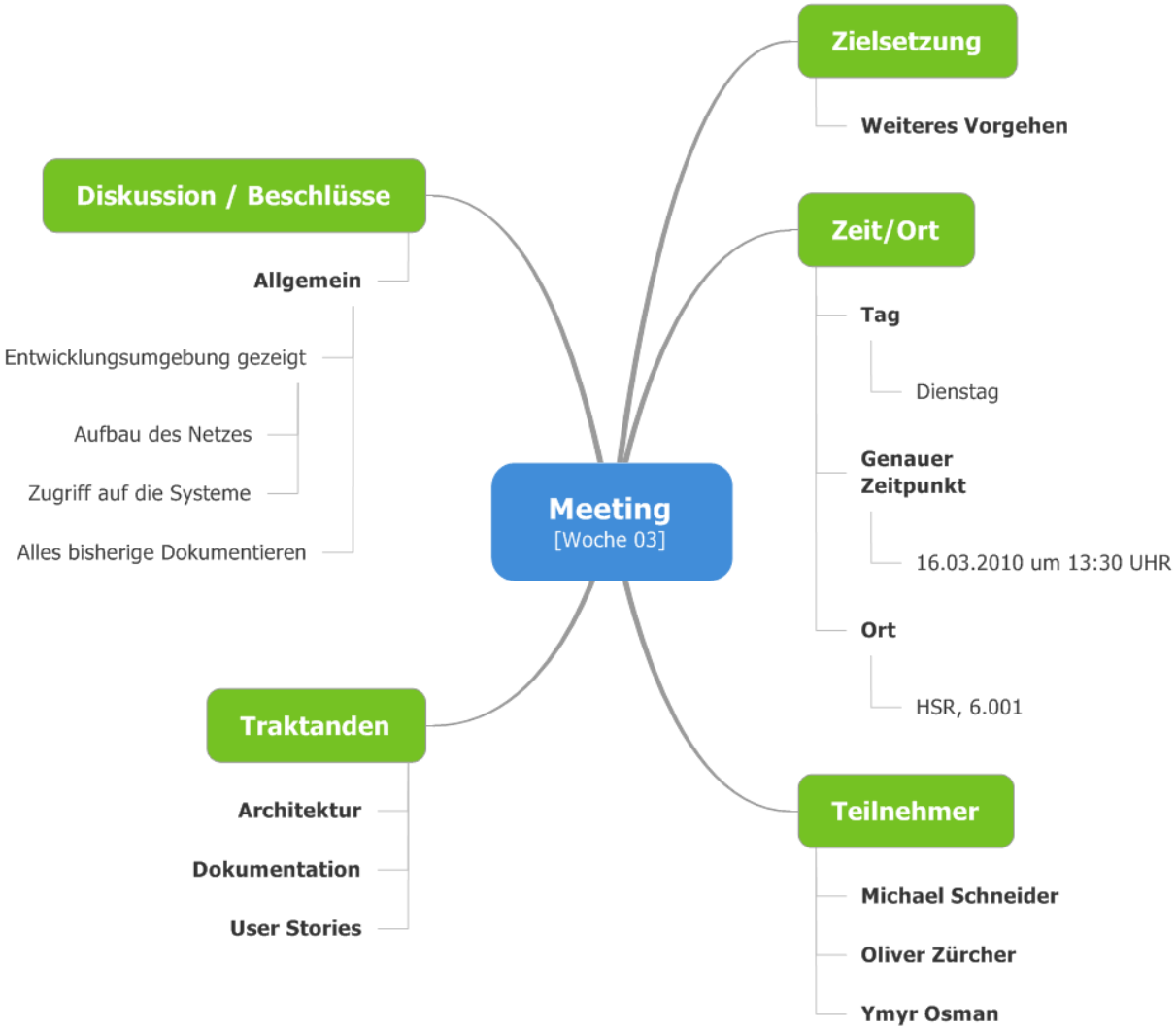
5.1 Woche 1



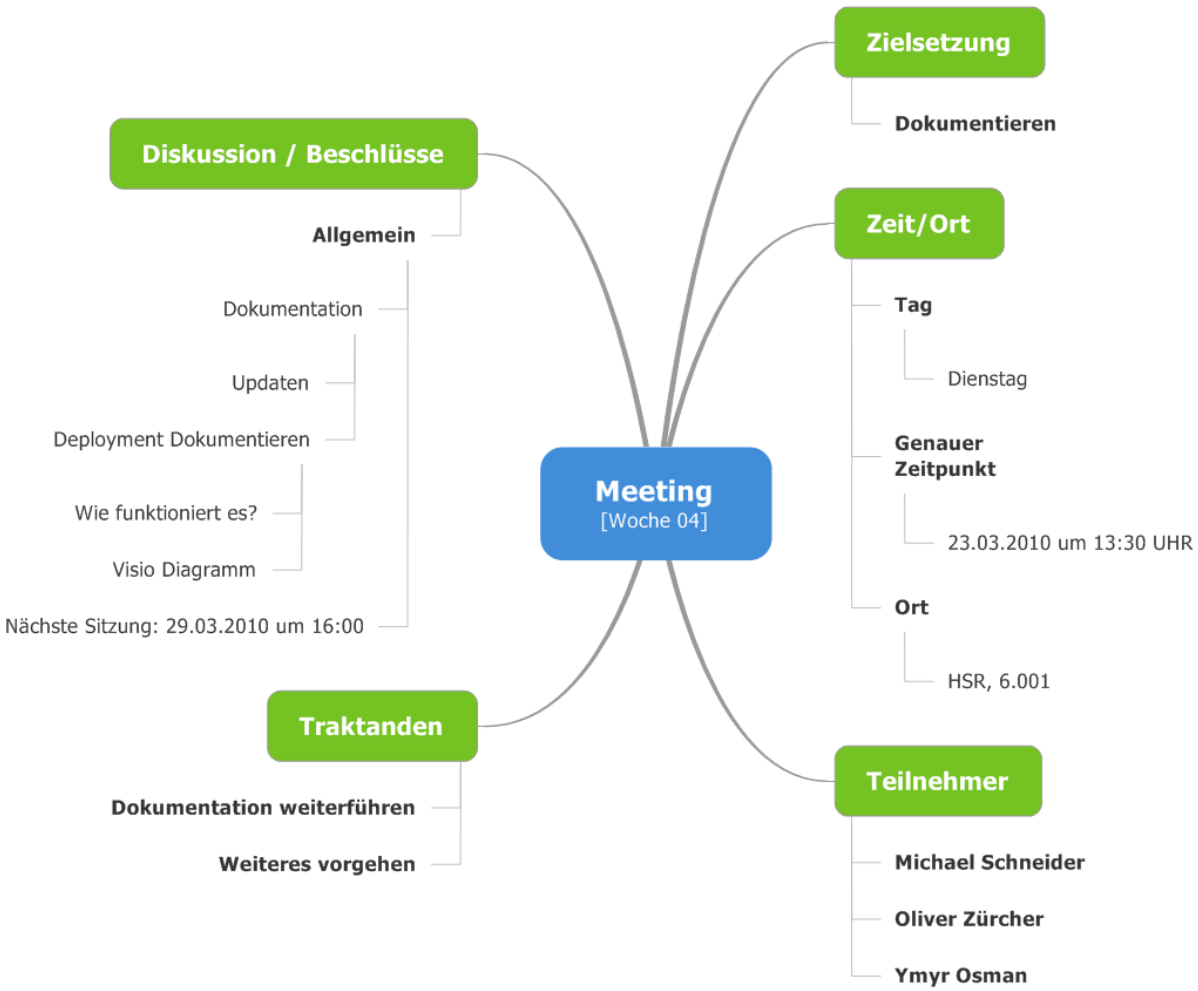
5.2 Woche 2



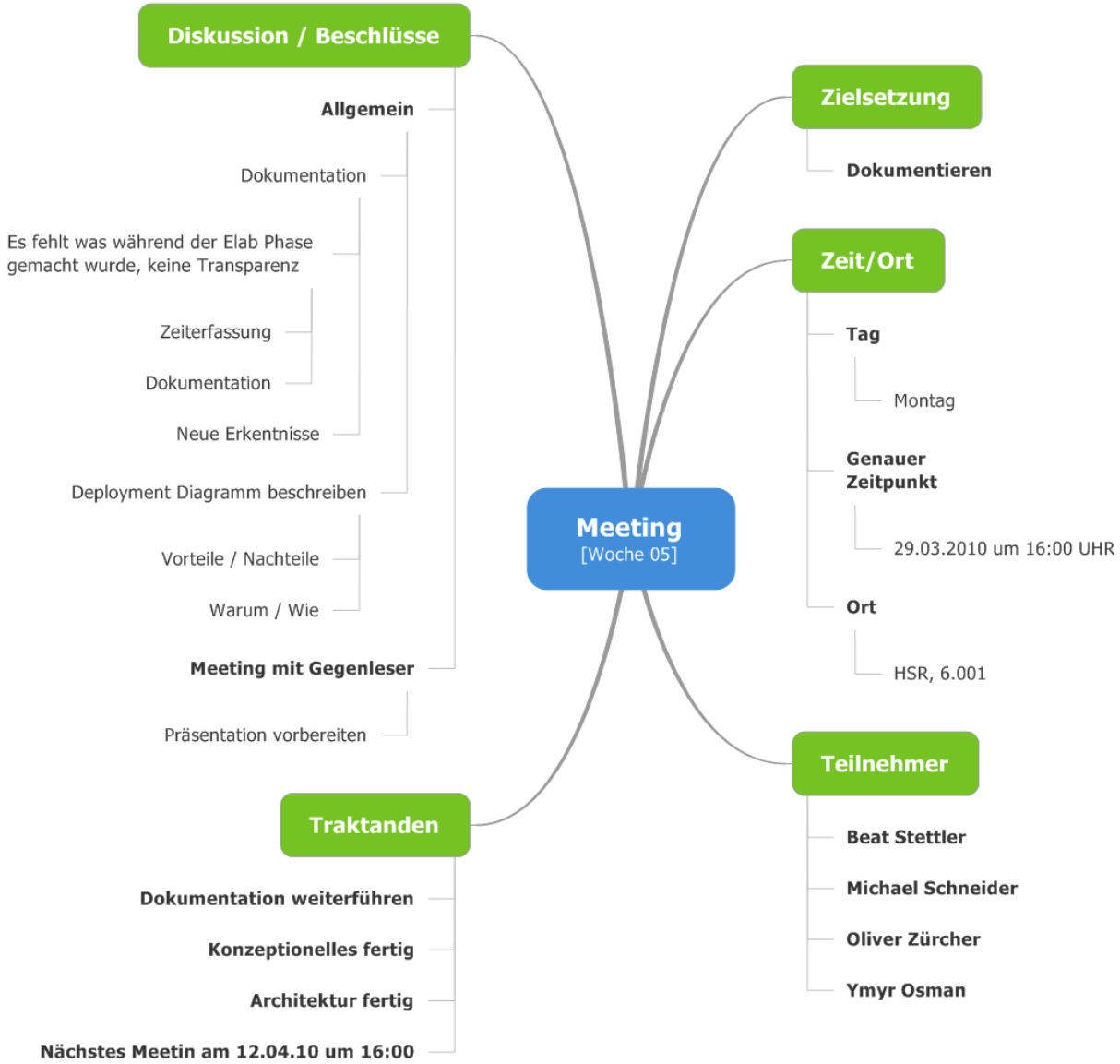
5.3 Woche 3


Meeting
[Woche 03]
Zielsetzung
Weiteres Vorgehen
Zeit/Ort
Tag
Dienstag
Genauer Zeitpunkt
16.03.2010 um 13:30 UHR
Ort
HSR, 6.001
Teilnehmer
Michael Schneider
Oliver Zürcher
Ymyr Osman
Diskussion / Beschlüsse
Allgemein
Entwicklungsumgebung gezeigt
Aufbau des Netzes
Zugriff auf die Systeme
Alles bisherige Dokumentieren
Traktanden
Architektur
Dokumentation
User Stories

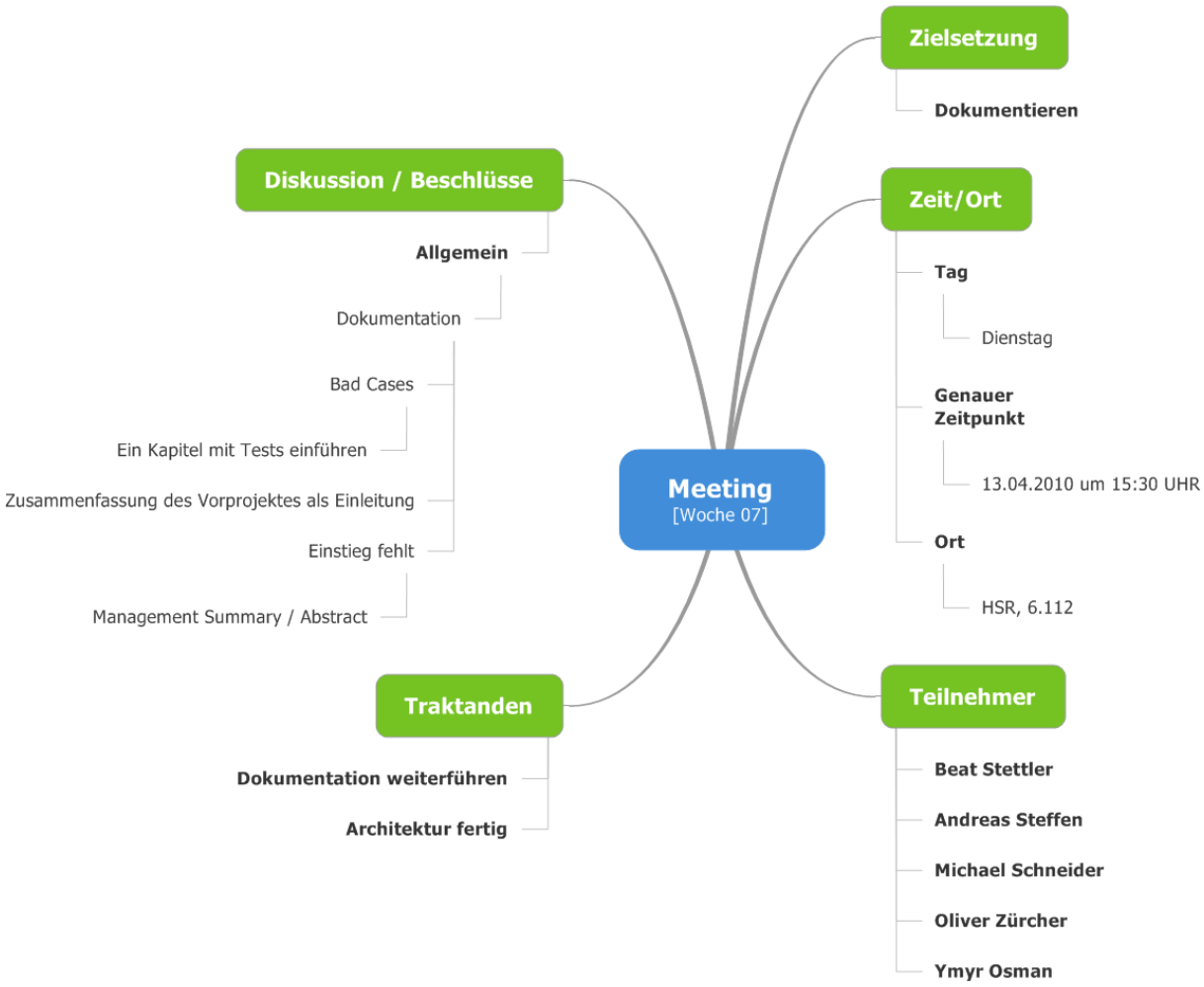
5.4 Woche 4


Meeting
[Woche 04]
Zielsetzung
Dokumentieren
Zeit/Ort
Tag
Dienstag
Genauer Zeitpunkt
23.03.2010 um 13:30 UHR
Ort
HSR, 6.001
Teilnehmer
Michael Schneider
Oliver Zürcher
Ymyr Osman
Diskussion / Beschlüsse
Allgemein
Dokumentation
Updaten
Deployment Dokumentieren
Wie funktioniert es?
Visio Diagramm
Nächste Sitzung: 29.03.2010 um 16:00
Traktanden
Dokumentation weiterführen
Weiteres vorgehen

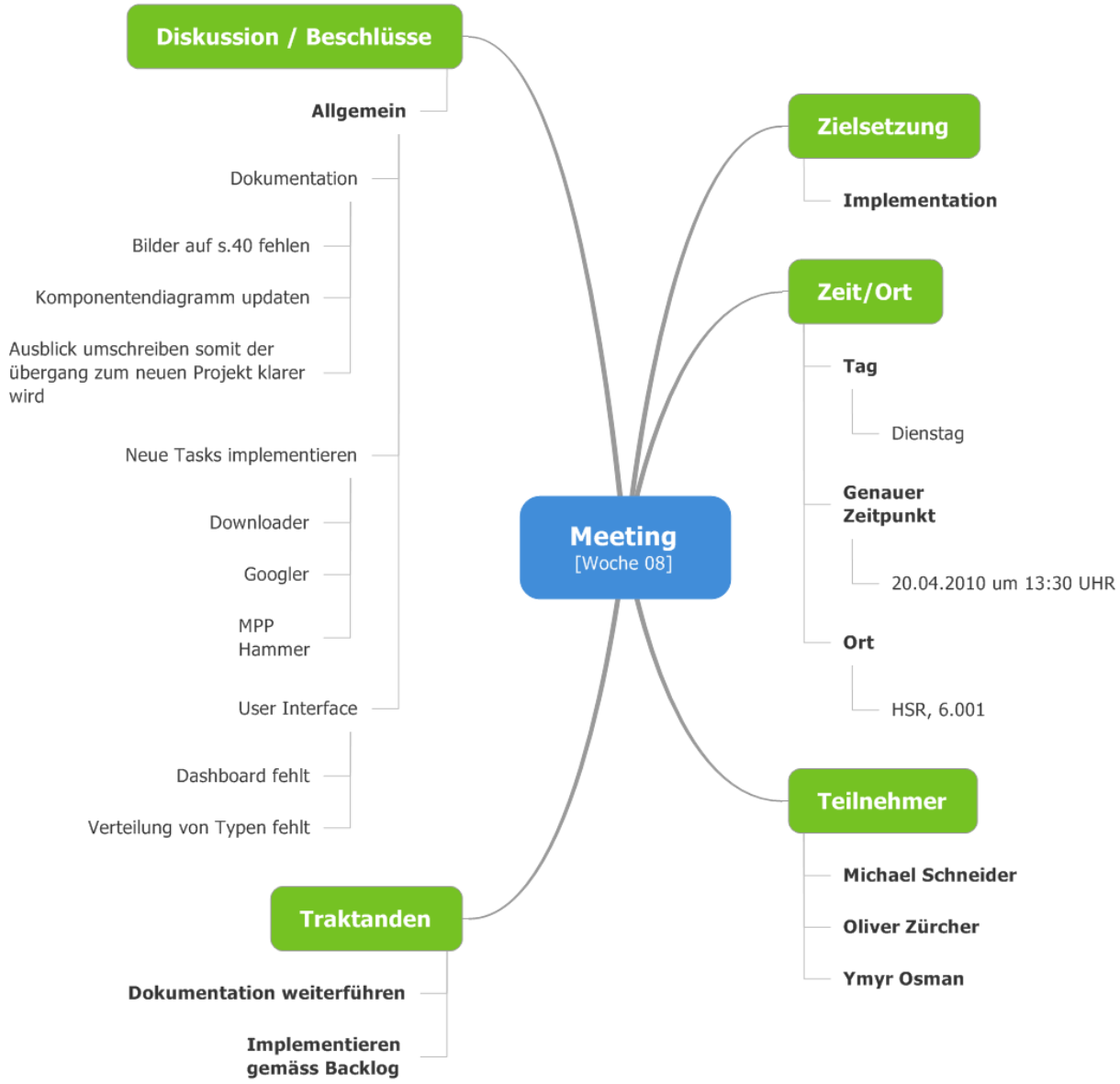
5.5 Woche 5



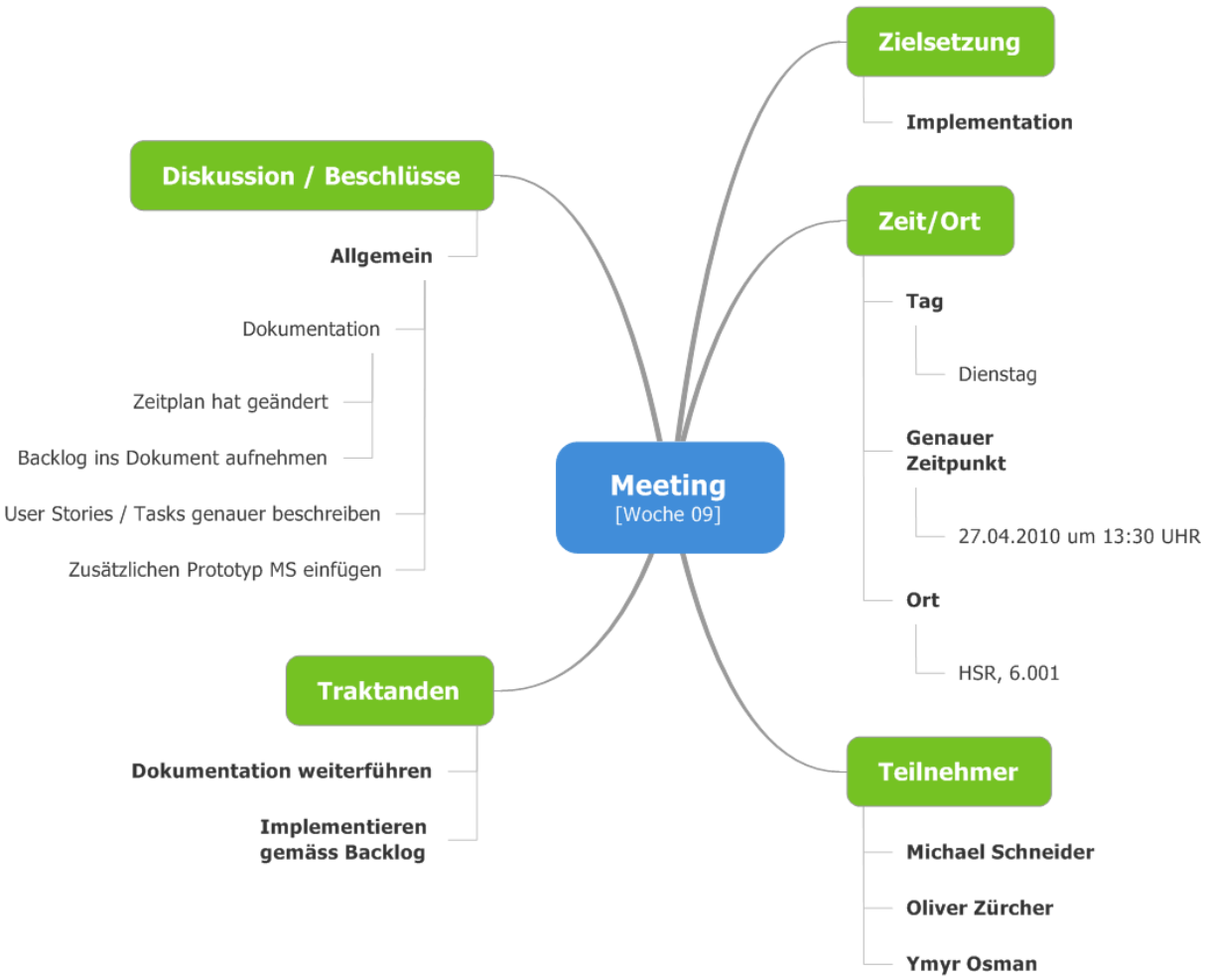
5.6 Woche 7


Meeting
[Woche 07]
Zielsetzung
Dokumentieren
Zeit/Ort
Tag
Dienstag
Genauer Zeitpunkt
13.04.2010 um 15:30 UHR
Ort
HSR, 6.112
Teilnehmer
Beat Stettler
Andreas Steffen
Michael Schneider
Oliver Zürcher
Ymyr Osman
Diskussion / Beschlüsse
Allgemein
Dokumentation
Bad Cases
Ein Kapitel mit Tests einführen
Zusammenfassung des Vorprojektes als Einleitung
Einstieg fehlt
Management Summary / Abstract
Traktanden
Dokumentation weiterführen
Architektur fertig

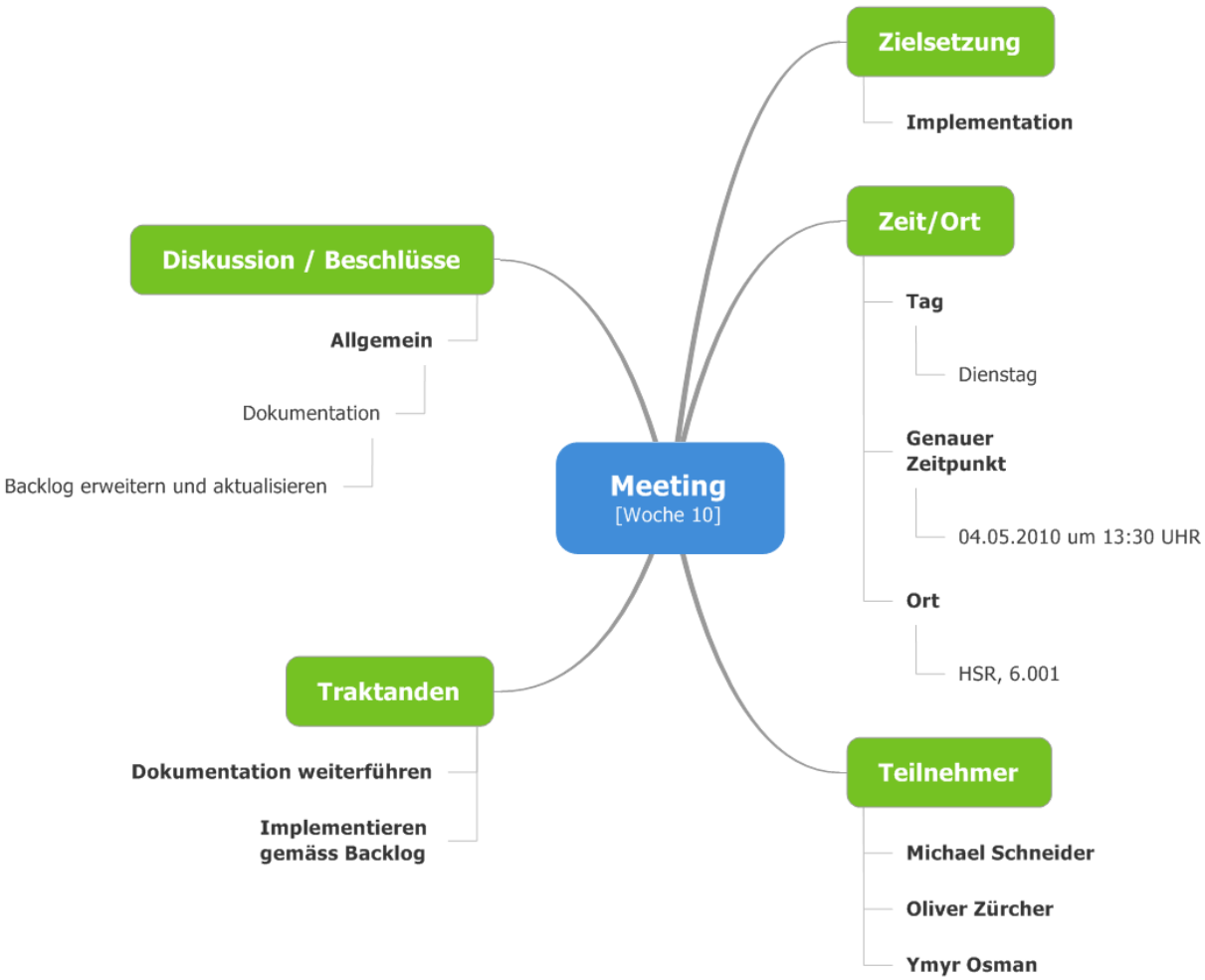
5.7 Woche 8


Meeting
[Woche 08]
Diskussion / Beschlüsse
Allgemein
Dokumentation
Bilder auf s.40 fehlen
Komponentendiagramm updaten
Ausblick umschreiben somit der übergang zum neuen Projekt klarer wird
Neue Tasks implementieren
Downloader
Googler
MPP Hammer
User Interface
Dashboard fehlt
Verteilung von Typen fehlt
Traktanden
Dokumentation weiterführen
Implementieren gemäss Backlog
Zielsetzung
Implementation
Zeit/Ort
Tag
Dienstag
Genauer Zeitpunkt
20.04.2010 um 13:30 UHR
Ort
HSR, 6.001
Teilnehmer
Michael Schneider
Oliver Zürcher
Ymyr Osman

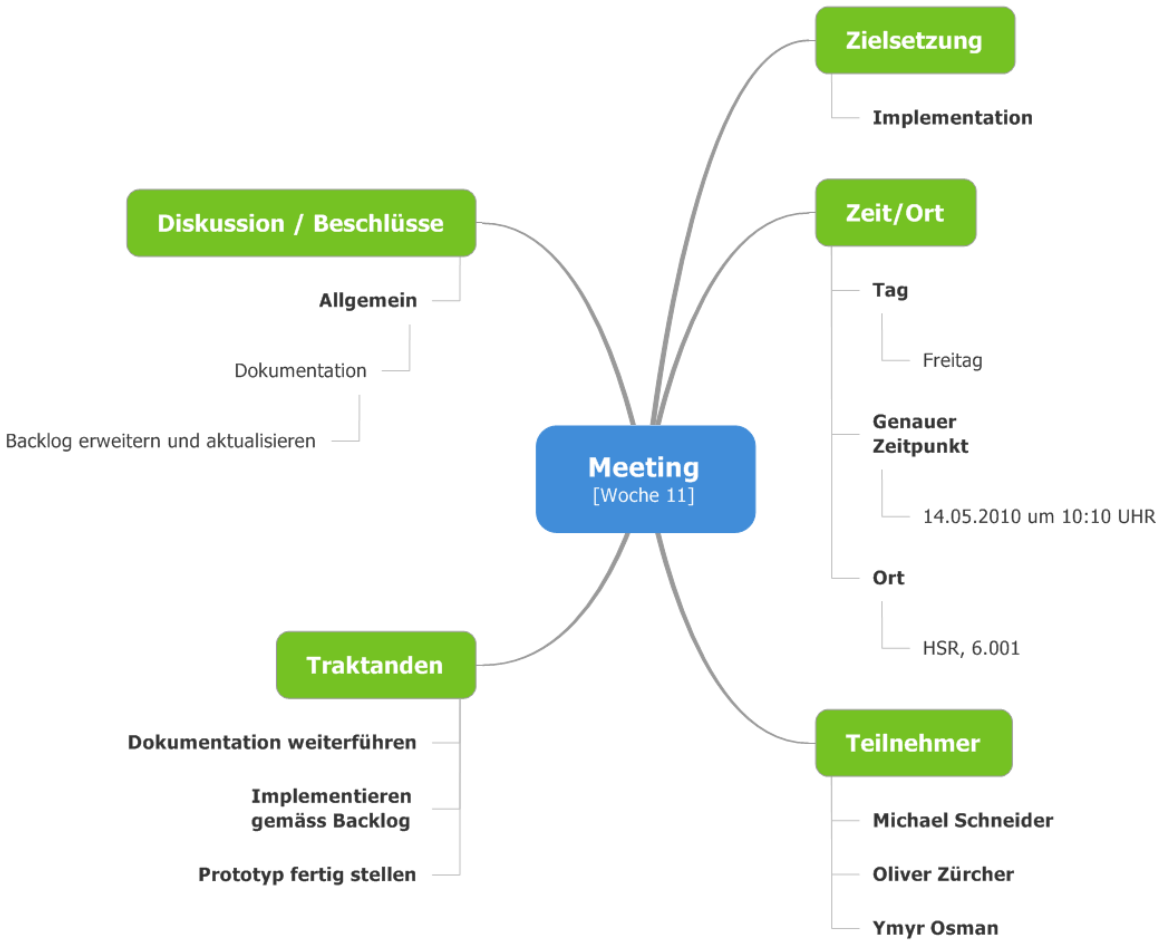
5.8 Woche 9


Meeting
[Woche 09]
Zielsetzung
Implementation
Zeit/Ort
Tag
Dienstag
Genauer Zeitpunkt
27.04.2010 um 13:30 UHR
Ort
HSR, 6.001
Teilnehmer
Michael Schneider
Oliver Zürcher
Ymyr Osman
Diskussion / Beschlüsse
Allgemein
Dokumentation
Zeitplan hat geändert
Backlog ins Dokument aufnehmen
User Stories / Tasks genauer beschreiben
Zusätzlichen Prototyp MS einfügen
Traktanden
Dokumentation weiterführen
Implementieren gemäss Backlog

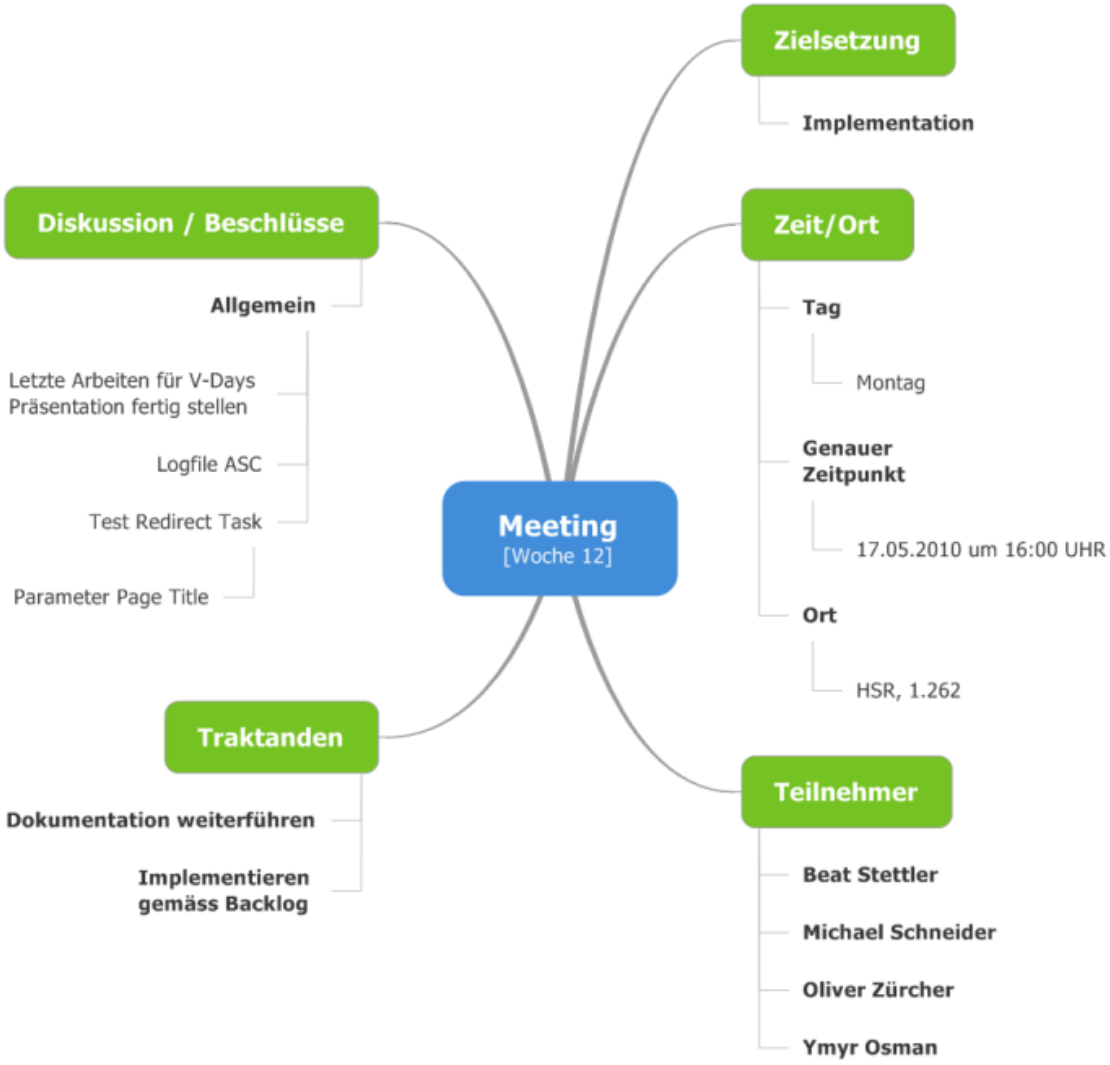
5.9 Woche 10


Meeting
[Woche 10]
Zielsetzung
Implementation
Zeit/Ort
Tag
Dienstag
Genauer Zeitpunkt
04.05.2010 um 13:30 UHR
Ort
HSR, 6.001
Teilnehmer
Michael Schneider
Oliver Zürcher
Ymyr Osman
Diskussion / Beschlüsse
Allgemein
Dokumentation
Backlog erweitern und aktualisieren
Traktanden
Dokumentation weiterführen
Implementieren gemäss Backlog

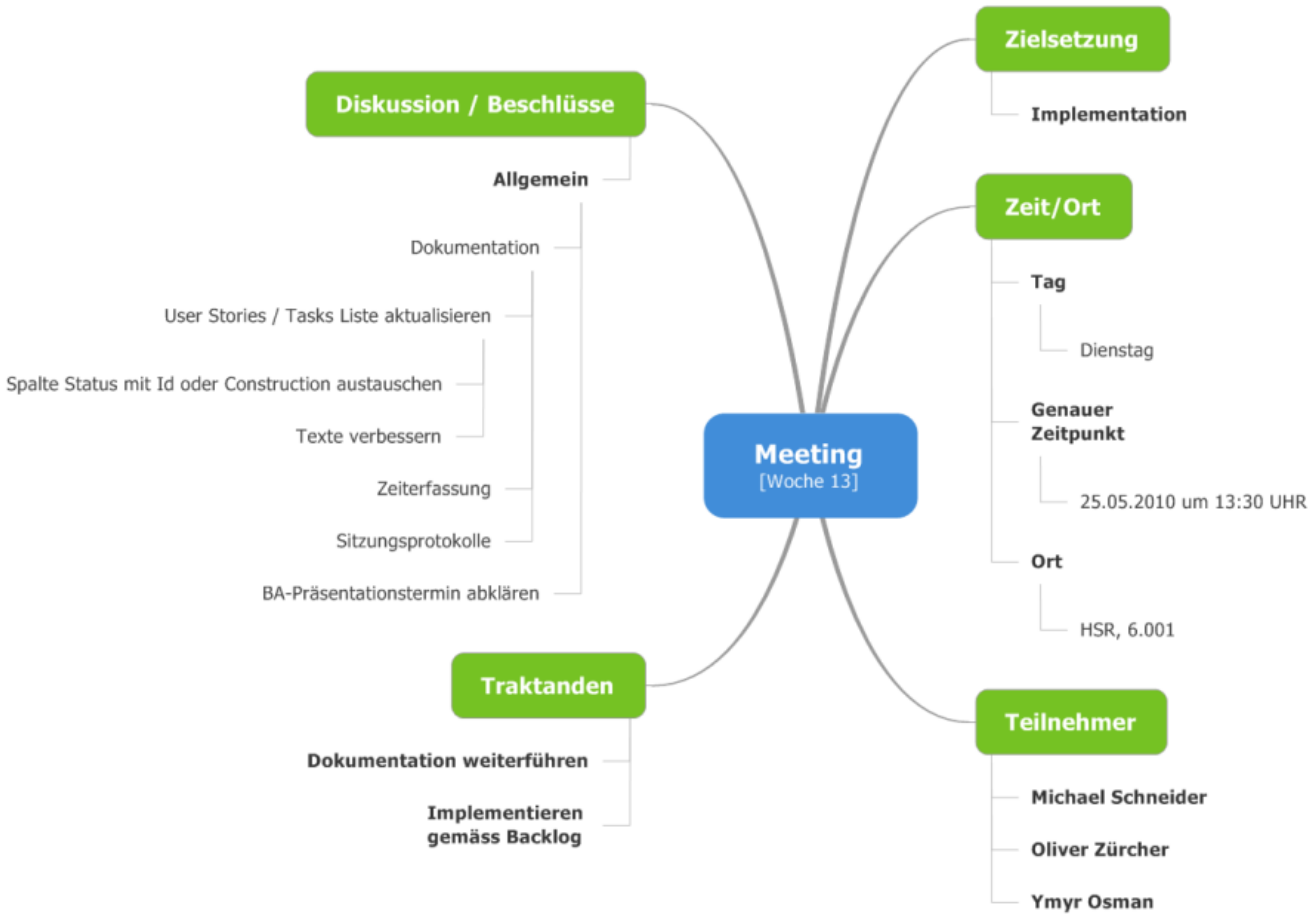
5.10 Woche 11


Meeting
[Woche 11]
Zielsetzung
Implementation
Zeit/Ort
Tag
Freitag
Genauer Zeitpunkt
14.05.2010 um 10:10 UHR
Ort
HSR, 6.001
Teilnehmer
Michael Schneider
Oliver Zürcher
Ymyr Osman
Diskussion / Beschlüsse
Allgemein
Dokumentation
Backlog erweitern und aktualisieren
Traktanden
Dokumentation weiterführen
Implementieren gemäss Backlog
Prototyp fertig stellen

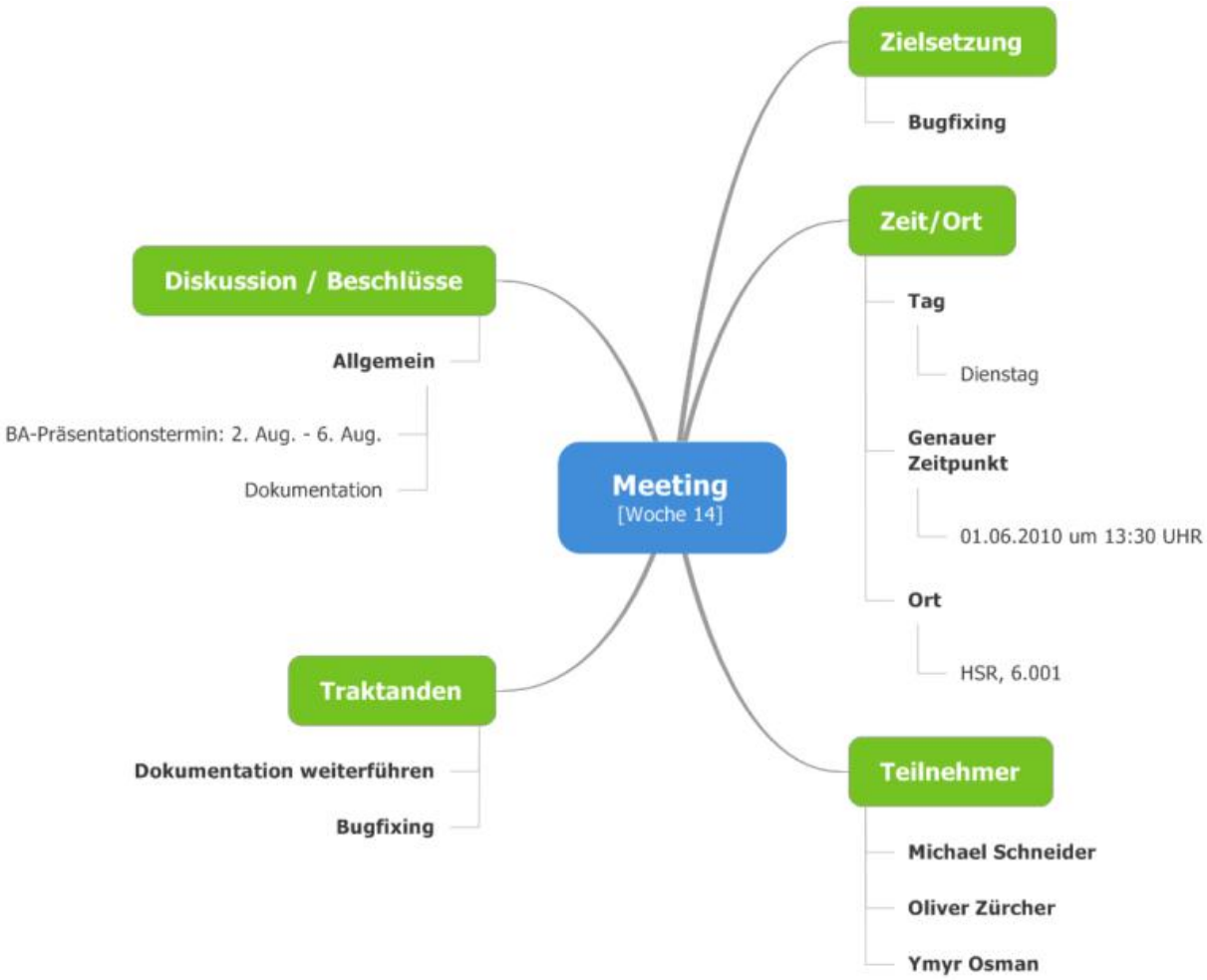
5.11 Woche 12


Meeting
[Woche 12]
Zielsetzung
Implementation
Zeit/Ort
Tag
Montag
Genauer Zeitpunkt
17.05.2010 um 16:00 UHR
Ort
HSR, 1.262
Teilnehmer
Beat Stettler
Michael Schneider
Oliver Zürcher
Ymyr Osman
Diskussion / Beschlüsse
Allgemein
Letzte Arbeiten für V-Days
Präsentation fertig stellen
Logfile ASC
Test Redirect Task
Parameter Page Title
Traktanden
Dokumentation weiterführen
Implementieren gemäss Backlog

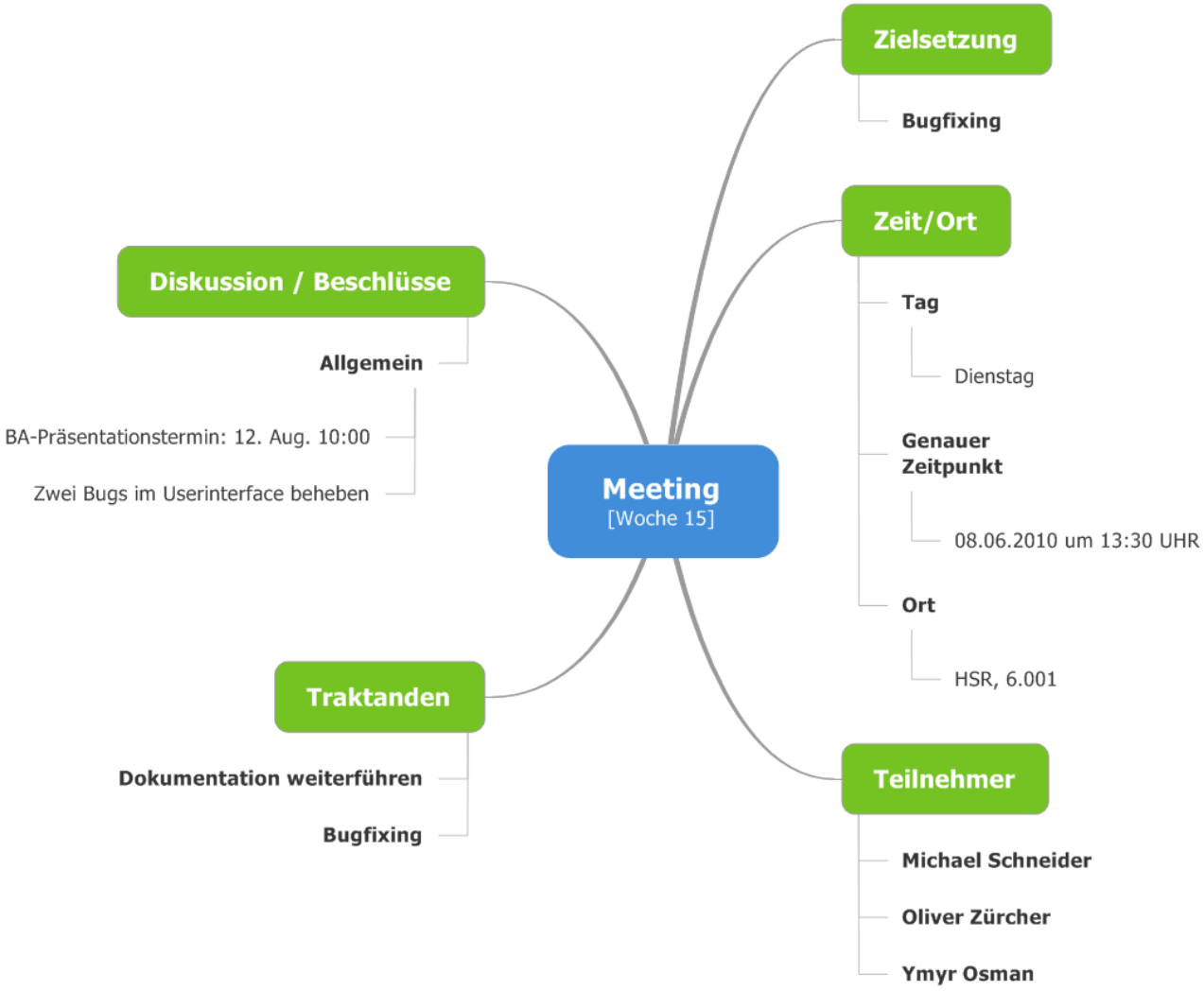
5.12 Woche 13


Meeting
[Woche 13]
Diskussion / Beschlüsse
Allgemein
Dokumentation
User Stories / Tasks Liste aktualisieren
Spalte Status mit Id oder Construction austauschen
Texte verbessern
Zeiterfassung
Sitzungsprotokolle
BA-Präsentationstermin abklären
Traktanden
Dokumentation weiterführen
Implementieren gemäss Backlog
Zielsetzung
Implementation
Zeit/Ort
Tag
Dienstag
Genauer Zeitpunkt
25.05.2010 um 13:30 UHR
Ort
HSR, 6.001
Teilnehmer
Michael Schneider
Oliver Zürcher
Ymyr Osman

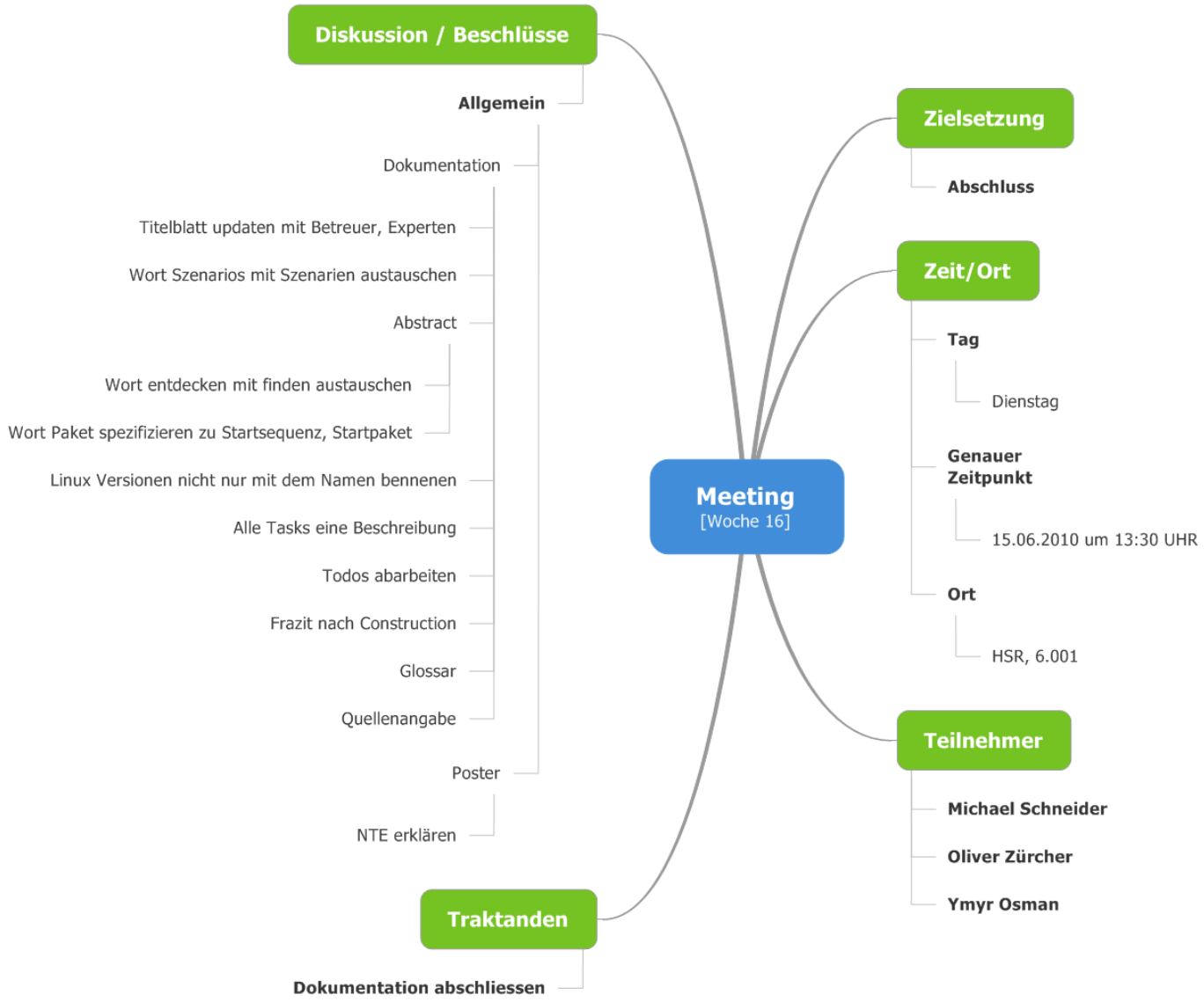
5.13 Woche 14


Meeting
[Woche 14]
Zielsetzung
Bugfixing
Zeit/Ort
Tag
Dienstag
Genauer Zeitpunkt
01.06.2010 um 13:30 UHR
Ort
HSR, 6.001
Teilnehmer
Michael Schneider
Oliver Zürcher
Ymyr Osman
Diskussion / Beschlüsse
Allgemein
BA-Präsentationstermin: 2. Aug. - 6. Aug.
Dokumentation
Traktanden
Dokumentation weiterführen
Bugfixing

5.14Woche 15


Meeting
[Woche 15]
Zielsetzung
Bugfixing
Zeit/Ort
Tag
Dienstag
Genauer Zeitpunkt
08.06.2010 um 13:30 UHR
Ort
HSR, 6.001
Teilnehmer
Michael Schneider
Oliver Zürcher
Ymyr Osman
Diskussion / Beschlüsse
Allgemein
BA-Präsentationstermin: 12. Aug. 10:00
Zwei Bugs im Userinterface beheben
Traktanden
Dokumentation weiterführen
Bugfixing

5.15 Woche 16


Meeting
[Woche 16]
Diskussion / Beschlüsse
Allgemein
Dokumentation
Titelblatt updaten mit Betreuer, Experten
Wort Szenarios mit Szenarien austauschen
Abstract
Wort entdecken mit finden austauschen
Wort Paket spezifizieren zu Startsequenz, Startpaket
Linux Versionen nicht nur mit dem Namen bennenen
Alle Tasks eine Beschreibung
Todos abarbeiten
Frazit nach Construction
Glossar
Quellenangabe
Poster
NTE erklären
Traktanden
Dokumentation abschliessen
Zielsetzung
Abschluss
Zeit/Ort
Tag
Dienstag
Genauer Zeitpunkt
15.06.2010 um 13:30 UHR
Ort
HSR, 6.001
Teilnehmer
Michael Schneider
Oliver Zürcher
Ymyr Osman

6. Zeiterfassung

6.1 Oliver Zürcher

Woche	Tag	Tätigkeit	Zeit	Zeit/Wo	Total	Soll
Woche 1	Dienstag	Konfigurationsmanagement TFS 2010	9.0			
	Dienstag	Meeting	0.5			
	Mittwoch	Projektplan	5.0			
	Donnerstag	Konfigurationsmanagement TFS 2010	8.0			
				22.5	**22.5**	20.0
Woche 2	Dienstag	Konfigurationsmanagement TFS 2010	9.0			
	Dienstag	Meeting	0.5			
	Dienstag	Projektplan	1.0			
	Dienstag	Arbeitscomputer aufsetzen	2.0			
	Mittwoch	Backuplösung TFS und SQL Server	3.0			
	Donnerstag	Hands On Workitems TFS2010	1.0			
	Donnerstag	TFS Timesheetversuch	0.5			
	Freitag	Dokumentation	4.5			
				21.5	**44.0**	40.0
Woche 3	Dienstag	Meeting	0.5			
	Dienstag	Solution erstellt & Deployment	2.0			
	Mittwoch	Domainanalyse	3.0			
	Mittwoch	Komponentendiagramm	2.0			
	Mittwoch	Datenbankmodell	3.0			
	Donnerstag	Entity Framework Hands on	4.0			
	Freitag	ASP.NET Hands on	6.5			
	Freitag	Deployment Diagramm	2.0			
				23.0	**67.0**	60.0
Woche 4	Dienstag	Meeting	0.5			
	Mittwoch	Testumgebung aufsetzen	8.0			
	Donnerstag	Testumgebung aufsetzen	8.0			
	Freitag	Testumgebung aufsetzen	4.0			
	Freitag	Deployment Diagramm	3.0			
				23.5	**90.5**	80.0

Woche 5	Dienstag	Meeting (mit Herr Stettler)	0.5			
	Dienstag	Dokumentation Technologieanalyse	3			
	Mittwoch	Technologieanalyse Kapitel Task	2			
	Mittwoch	Doku Captions erstellt, IV formatiert	1			
	Mittwoch	Doku Textüberarbeitung	2			
	Donnerstag	Layer Diagramm	4			
	Freitag	Prototyp Debugging	3.5			
	Freitag	Dokumentation	2.0			
				18.0	**108.5**	100.0
Woche 6	Mittwoch	Prototyp Debugging	6			
	Donnerstag	Prototyp Debugging	6			
	Donnerstag	SQL Server Konfiguration/ConString	2.5			
	Freitag	Reflection Debugging	8			
				22.5	**131.0**	120.0
Woche 7	Dienstag	Abwesenheit Techdays	0			
	Mittwoch	Abwesenheit Techdays	0			
	Donnerstag	Deploymentskript für Services	6			
	Donnerstag	Deploymentskripts (WinSCP)	3			
	Freitag	Weiterarbeit Architekturprototyp	4			
	Freitag	Doku. SAD, Zeitplan Änderung	4			
				17.0	**148.0**	140.0
Woche 8	Dienstag	Weiterarbeit Architekturprototyp	7.0			
	Mittwoch	Dokumentation	3.0			
	Donnerstag	SQL Server Konfiguration/ConString	3.0			
	Freitag	Dokumentation	2.0			
				15.0	**163.0**	160.0
Woche 9	Dienstag	OR Mapping auf Code First geändert	4.0			
	Mittwoch	OR Mapping auf Code First geändert	6.0			
	Donnerstag	Tasks implementiert	3.0			
	Donnerstag	Dokumentation	1.0			
	Freitag	Task API Dokumentation	5.0			
				19.0	**182.0**	180.0

Woche 10	Dienstag	Task Reflection	3.0			
	Dienstag	Task API Dokumentation	1.0			
	Mittwoch	Task API Dokumentation	2.0			
	Mittwoch	Task in LoadGenerator einbetten	8.0			
	Donnerstag	TeamSpec Hands On	4.0			
	Donnerstag	Workitems in Word integriert	3.0			
				21.0	**203.0**	200.0
Woche 11	Montag	Bug fixing Mono Timeout Probleme	4.0			
	Montag	Bug fixing Mono Timeout Probleme	4.0			
	Dienstag	MPPHammer implementieren	3.0			
	Dienstag	VM Verteilung / MAC Generator	3.0			
	Mittwoch	Integration Workitems in Word	3.0			
	Donnerstag	Integration Workitems in Word	4.0			
				21.0	**224.0**	220.0
Woche 12	Montag	Taskausführung via Reflection	4.0			
	Dienstag	Taskausführung via Reflection	3.0			
	Mittwoch	.NET Remoting Debugging	4.0			
	Donnerstag	.NET Remoting Debugging	4.0			
	Freitag	Refactorings im Code / Codereview	3.0			
				18.0	**242.0**	240.0
Woche 13	Montag	.NET Remoting Debugging	1.0			
	Montag	Backend Reporting	4.0			
	Dienstag	Backend Reporting	6.0			
	Mittwoch	Backend Reporting	6.0			
	Donnerstag	Backend Reporting	4.0			
	Freitag	Dokumentation	4.0			
				25.0	**267.0**	260.0
Woche 14	Montag	Googler Task implementieren	4.0			
	Montag	Googler Task deployen/testen	2.0			
	Montag	Googler Task bugfixing	1.0			
	Dienstag	Dokumentation	3.0	**20.0**	**287.0**	280.0

	Dienstag	Refactorings Googler Task	3.0			
	Mittwoch	Vorlage Word	3.0			
	Mittwoch	Threading im Backend verbessert	4.0			
Woche 15	Montag	Code Refactorings	7.0			
	Dienstag	Dokumentation Fehlerkorrektur	4.0			
	Dienstag	Bugfixing Kommunikation mit WI	4.0			
	Mittwoch	Bugfixing Kommunikation mit WI	8.0			
	Donnerstag	Bugfixing Kommunikation mit WI	4.0			
	Donnerstag	Installationsscripts	6.0			
	Freitag	Installationsscripts	5.0			
	Freitag	Dokumentation	4.0			
				42.0	**329.0**	320.0
Woche 16	Montag	Dokumentation	9.0			
	Dienstag	Dokumentation	10.0			
	Mittwoch	Dokumentation	10.0			
	Donnerstag	Publish	11.0			
				40.0	**369.0**	360.0

6.2 Ymyr Osman

Woche	Tag	Tätigkeit	Zeit	Zeit/Wo	Total	Soll
Woche 1	Dienstag	Meeting	0.5			
	Dienstag	Konfigurationsmanagement TFS	9.0			
	Mittwoch	Projetkplan	5.5			
	Donnerstag	Konfigurationsmanagement TFS	5.0			
				20.0	**20.0**	20.0
Woche 2	Dienstag	Meeting	0.5			
	Dienstag	Projetkplan	2.0			
	Mittwoch	Backup	1.5			
	Mittwoch	Konfigurationsmanagement TFS	3.0			
	Mittwoch	Arbeitsplatz aufgesetzt	1.5			
	Mittwoch	TFS Scrum Einarbeitung	3.0			
	Mittwoch	Hands On TFS Workitems	3.0			
	Donnerstag	Dokumentation	3.0			
	Freitag	Dokumentation	2.5			
				20.0	**40.0**	40.0
Woche 3	Dienstag	Meeting	0.5			
	Dienstag	Webinterface ASP aufgesetzt	1.5			
	Mittwoch	Domainanalyse	2.0			
	Mittwoch	Component	4.0			
	Mittwoch	Datenbankdesign	4.0			
	Donnerstag	Entity Framework Hands On	4.0			
	Freitag	ASP.NET Hands On	6.5			
	Freitag	Deployment Diagramm	1.0			
				23.5	**63.5**	60.0
Woche 4	Dienstag	Meeting	0.5			
	Dienstag	ASP.NET Hands On	6.5			
	Mittwoch	Aufsetzen der Testumgebung	8.0			
	Donnerstag	Aufsetzen der Testumgebung	8.0			
	Freitag	Aufsetzen der Testumgebung	4.0			
	Freitag	Dokumentation	1.0			
				28.0	**91.5**	80.0
Woche 5	Montag	ASP.NET Erarbeitung	4.0			
	Montag	Webinterface	3.0	**22.0**	**113.5**	100.0

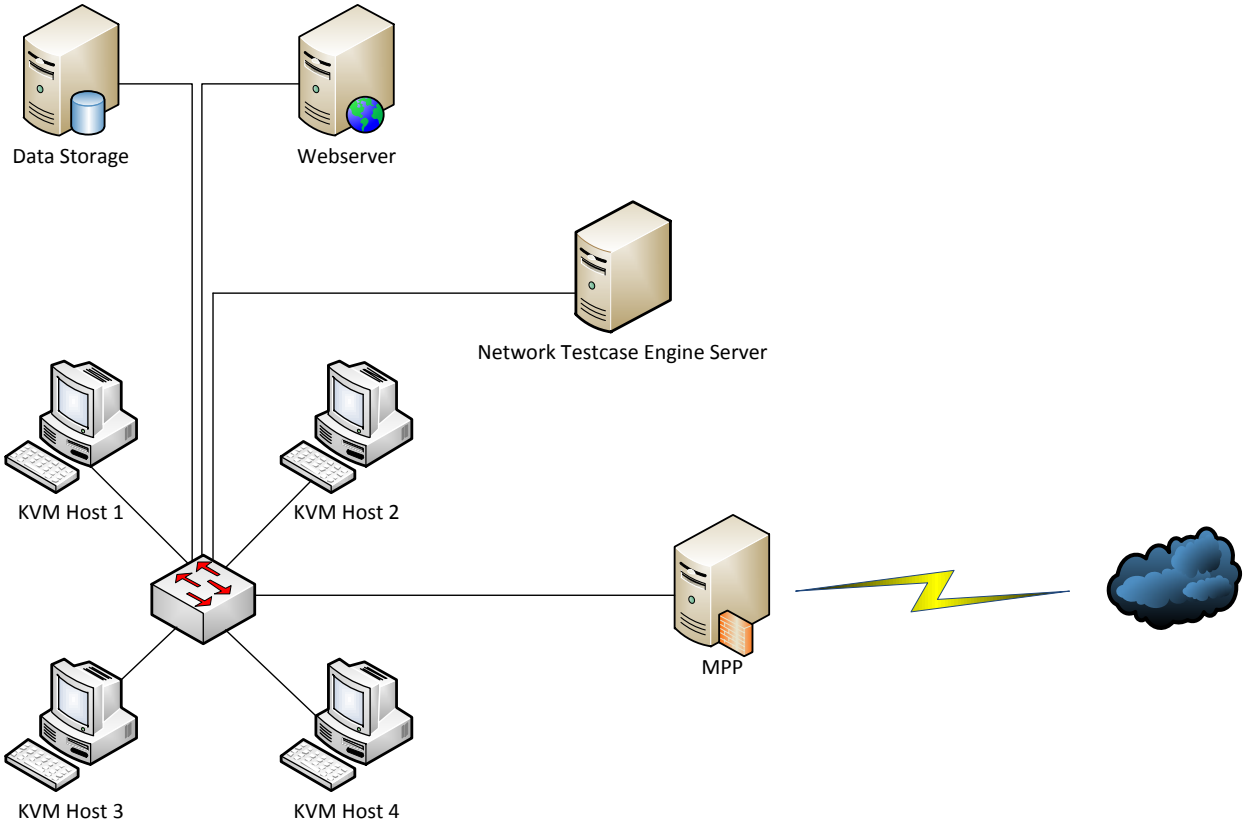
	Dienstag	Meeting (mit Herr Stettler)	0.5			
	Dienstag	Dokumentation Technologieanalyse	5.0			
	Dienstag	Deployment	2.5			
	Dienstag	Webinterface	3.0			
	Mittwoch	Webinterface	4.0			
Woche 6	Dienstag	ExtJs Erarbeitung	2.0			
	Mittwoch	Webinterface	7.0			
	Donnerstag	Webinterface	5.0			
	Freitag	Deployment	1.5			
	Freitag	Webinterface	5.0			
				20.5	**134.0**	120.0
Woche 7	Dienstag	Abwesenheit Techdays				
	Mittwoch	Abwesenheit Techdays				
	Donnerstag	Webinterface	7.0			
	Donnerstag	ExtJs Erarbeitung	1.0			
	Freitag	Deployment	1.0			
	Freitag	Webinterface	5.0			
	Samstag	Dokumentation	3.5			
				17.5	**151.5**	140.0
Woche 8	Dienstag	Erarbeitung Datenmodell	2.0			
	Dienstag	ExtJs Erarbeitung	2.0			
	Mittwoch	Webinterface	5.0			
	Mittwoch	ExtJs Erarbeitung	2.0			
	Donnerstag	Dokumentation	2.0			
	Freitag	Webinterface	5.0			
				18.0	**169.5**	160.0
Woche 9	Dienstag	Tasks Implemnetieren	3.5			
	Mittwoch	Tasks Implemnetieren	7.0			
	Donnerstag	Tasks Implemnetieren	3.5			
	Freitag	Tasks Implemnetieren	3.5			
	Sonntag	Dokumentation	2.0			
				19.5	**189.0**	180.0

Woche 10	Dienstag	Tasks Implemnetieren	3.0			
	Mittwoch	Tasks Implemnetieren	8.0			
	Donnerstag	Tasks Implemnetieren	4.5			
	Freitag	Tasks Implemnetieren	2.0			
	Samstag	Tasks Implemnetieren	2.5			
				20.0	**209.0**	200.0
Woche 11	Dienstag	Tasks Implemnetieren	3.5			
	Mittwoch	Tasks Implemnetieren	7.0			
	Donnerstag	Tasks Implemnetieren	2.0			
	Freitag	Tasks Implemnetieren	2.0			
	Samstag	Tasks Implemnetieren	2.5			
	Sonntag	Dokumentation	2.0			
				19.0	**228.0**	220.0
Woche 12	Dienstag	Tasks Implemnetieren	2.5			
	Mittwoch	Tasks Implemnetieren	7.5			
	Donnerstag	Tasks Implemnetieren	4.5			
	Freitag	Tasks Implemnetieren	4.5			
	Sonntag	Dokumentation	2.0			
				21.0	**249.0**	240.0
Woche 13	Dienstag	Tasks Implemnetieren	3.5			
	Mittwoch	Tasks Implemnetieren	8.0			
	Donnerstag	Tasks Implemnetieren	3.0			
	Freitag	Tasks Implemnetieren	4.0			
	Samstag	Dokumentation	3.0			
				21.5	**270.5**	260.0
Woche 14	Dienstag	Tasks Implemnetieren	4.0			
	Mittwoch	Tasks Implemnetieren	8.0			
	Donnerstag	Tasks Implemnetieren	4.0			
	Freitag	Dokumentation	2.0			
	Freitag	Dokumentation	2.0			
				20.0	**290.5**	280.0

Woche 15	Montag	Bugfixing	4.5			
	Montag	Refactoring	4.5			
	Dienstag	Bugfixing	6.5			
	Dienstag	Refactoring	2.5			
	Mittwoch	Refactoring	1.0			
	Mittwoch	Bugfixing	7.0			
	Donnerstag	Refactoring	0.5			
	Donnerstag	Dokumentation	6.5			
	Freitag	Dokumentation	6.0			
	Freitag	Bugfixing	2.0	**41.0**	**331.5**	320.0
Woche 16	Montag	Dokumentation	8.0			
	Dienstag	Dokumentation	8.0			
	Mittwoch	Dokumentation	8.0			
	Donnerstag	Publishing	8.0			
				32.0	**363.5**	360.0

7. Installationsanleitung

In den folgenden Kapitel wird die Installation der Komponenten im folgenden Bild erläutert.



7.1 KVM Hosts (Nodes)

Die Installation eines KVM Hosts gestaltet sich sehr einfach. Nach einer Grundinstallation von Ubuntu 10.4 „Lucid Lynx“ (Desktop Oberfläche wird nicht benötigt), muss lediglich das node_install.sh Skript von der CD als root ausgeführt werden. Kurz zusammengefasst erledigt das Skript folgende Dinge:

1. Installiert mit apt bridge-utils, kvm, mono, mono libs und sudo
2. Erstellt Benutzer „captive“ mit homedir /home/captive
3. Richtet ein Bridge Interface br0 ein
4. Installiert kvm-ifup/ifdown Skripte
5. Kopiert notwendige Assemblies nach /home/captive
6. Kopiert das Basis Image (base.img) nach /home/captive

Danach ist die Node bereit. Mit dem Befehl mono lässt sich NTE.VMRemoteControl.exe starten. Folgendes Script kann in /etc/network/if-up.d/VMRemoteControl abgelegt werden, damit die VMRemoteControl Anwendung automatisch nach dem Bootvorgang startet.

```
#! /bin/sh

set -e

if [ "$IFACE" = lo ]; then
        exit 0
fi

# Only run from ifup.
if [ "$MODE" != start ]; then
        exit 0
fi

cd /home/captive
mono NTE.VMRemoteControl.exe &> VMRemoteControl.log

exit 0
```

7.2 Data Storage

Als Datenbankserver wurde der Microsoft SQL Server 2008 eingesetzt. Mit der DBCreate.exe Anwendung auf der CD wird die Datenbank mit den nötigen Tabellen erstellt. Die Datenbank heisst „NetworkTestcaseEngine“.
Alle Programme laufen im Security Mode „integrated“. Es muss also sichergestellt sein, dass der Benutzer, unter welchem die Assemblies laufen, genügend Rechte auf dem SQL Server hat. Dies gilt für NTE.Remoting.exe, NTE.Service.exe und DBCreate.exe.

7.3 Webserver

Als Webserver genügt ein IIS 7 Webserver mit der standard Konfiguration. Die Applikation muss über den Pfad /NTE erreichbar sein. Die Webapplikation ist im Ordner inetpub auf der CD zu finden.

7.4 Network Testcase Engine (Services)

Die beiden Services NTE.Service.exe und NTE.Reporting.exe werden in der Regel auf der selben Maschine wie die Webapplikation installiert. Für die Installation der Services wird das Tool installutil verwendet:

```
C:\repo>installutil.exe NTE.Service.exe
C:\repo>installutil.exe NTE.Reporting.exe
```

Mit diesen beiden Befehlen werden die Services installiert. Danach können die Services mit dem MMC Snap-In „services.msc“ gestartet werden. Zu beachten ist, dass die Services per default mit dem Benutzer „Network Service“ gestartet werden. Dieser benötigt somit genügend Rechte auf dem Datenbank Server.

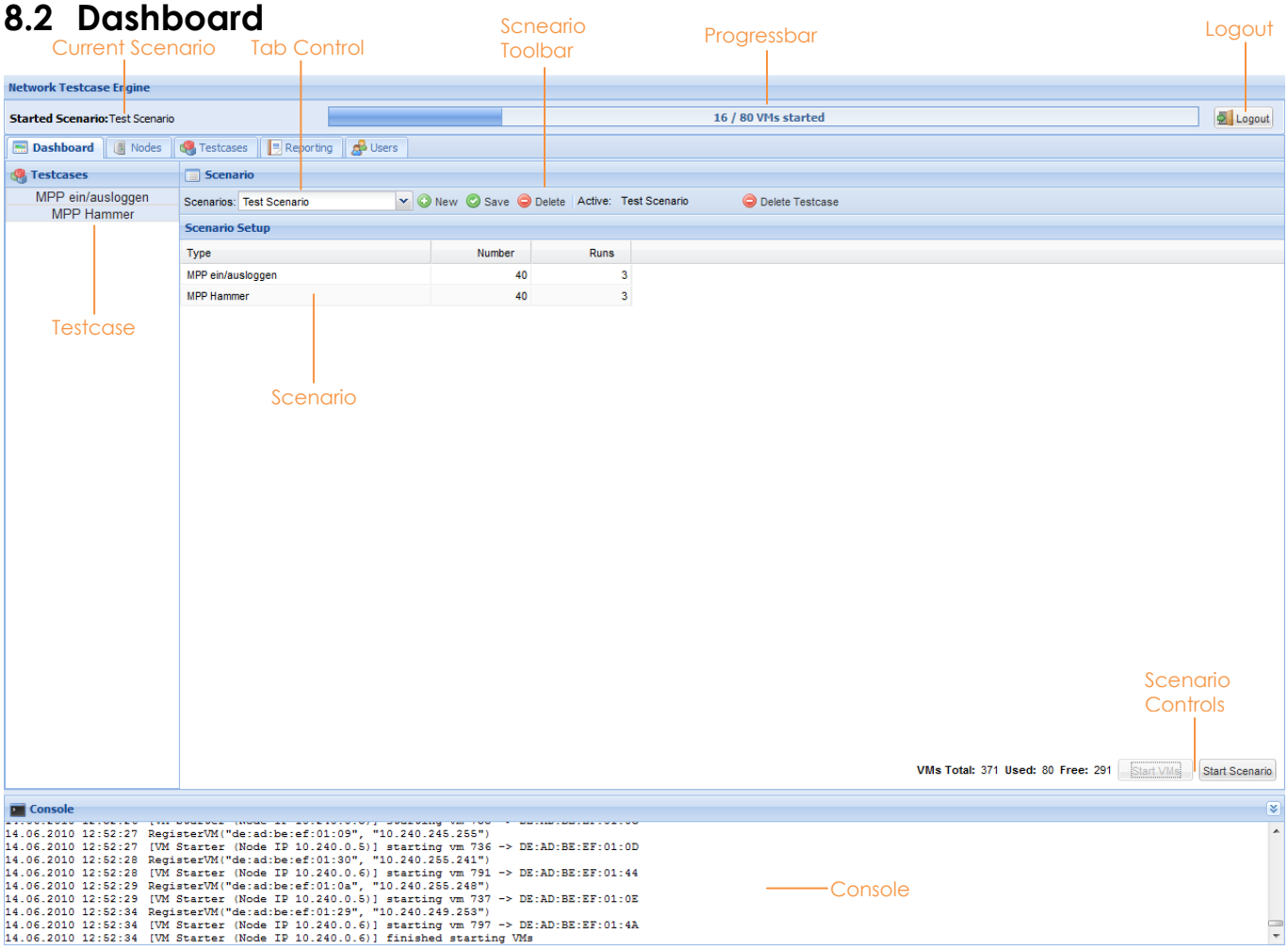
8. Bedienungsanleitung

8.1 Quickstart

Dieser Quicktart zeigt die nötigen Schritte auf um ein Beispielszenario durchzuführen.

1. Anmelden als Benutzer auf der Webapplikation
2. In das Testcase Tab wechseln und neuen Testcase erstellen
3. Mittels Drag und Drop gewünschte Tasks hinzufügen, Taskoptionen einstellen und speichern
4. In das Nodes Tab wechseln und Node Discovery ausführen
5. Tree „refreshen" und sehen ob Nodes vorhanden sind → evtl Refresh wiederholen
6. In das Dashboard Tab wechseln und neues Szenario erstellen
7. Mittels Drag und Drop gewünschte Testcases hinzufügen
   Number: *Anzahl VMs, welche diesen Testcase ausführen*
   Runs: *Legt fest, wievielmal der Testcase auf der virtuellen Maschine wiederholt wird*
8. Start VMs drücken und warten bis gwünschte Anzahl VMs vorhanden sind
9. Im Nodes Tab werden die fertig aufgestarteten VMs mit zugewiesenem Testcase und IP in der Treeview angezeigt (refresh Button drücken um die Treeview zu aktualisieren)
10. Wenn die VMs gestartet sind, „Start Szenario". Der Fortschritt des Szenariodurchlaufes ist in der Progressbar ganz oben ersichtlich
11. Im Reporting Tab kann ein Bericht zum Szenariodurchlauf angezeigt werden. Wenn Tasks fehlschlagen, sind weitere Informationen dazu auf der Detailseite eines Reports zu finden
12. Doppelclick auf Report um einen detailierten Stand des Szenarios zu sehen

8.2 Dashboard



8.2.1 Tab Control

Die Funktionen wurden in 5 Gruppen aufgeteilt und werden im UI als 5 Tabs dargestellt.

8.2.2 Testcases

Auf der linken Seite werden alle erstellten Testcases aufgelistet. Diese können rechts in das Scenario Setup Feld gezogen werden, um sie in einem Szenario einzubetten.

8.2.3 Scenario

Hier werden die einzelnen Testcases welche im Szenario enthalten sind angezeigt. Wenn ein neues Szenario erstellt wird ist diese Tabelle leer.
Um einen Testcase einem Szenario hinzuzufügen, kann man den Testcase mittels Drag n Drop in die Tabelle ziehen.
Um einen Testcase aus einem Szenario wieder zu löschen, kann dieser in der Tabelle markiert werden und in der „Scenario Toolbar“ mittels „Delete Testcase“ gelöscht werden.

8.2.4 Scenario Toolbar

8.2.4.1 Scenario Combobox

In der Combobox werden alle vorhandenen Szenarien aufgelistet. Um eines auszuwählen kann per Mausclick die Liste angezeigt werden oder der Name eingetippt werden. Nach dem Auswählen wird das Szenario automatisch geladen.

8.2.4.2 New

Beim drücken diese Buttons wir ein neues Szenario initalisiert. Es wird zufälliger ein Name gewählt, welcher aber über das Input Feld geändert werden kann.

8.2.4.3 Save

Dieser Button speichert alle Änderungen welche gemacht wurden am Szenario. Alle Änderungen werden erst übernommen, nachdem dieser Knopf gedrückt wurde.

8.2.4.4 Delete

Diese Aktion löscht das ausgewählt Szenario. Beim Löschen ist zu beachten, dass alles was mit dem Szenario in Verbindung steht auch gelöscht wird.

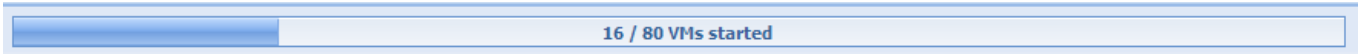
8.2.5 Active

Hier wird das aktuell gwählte Szenario angezeigt, damit vom Benutzer gesehen kann auf welchem Szenario er momentan die Änderungen vornimmt und nicht ausversehen am falschen Szenario arbeitet und dies abspeichert.

8.2.6 Scenario Controls

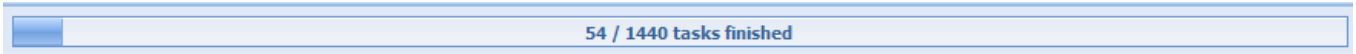
8.2.6.1 Start VMs

Mittels „StartVms“ werden die benötigten VMs zum Szenario augestartet. Zuerst muss allerdings eine „Node Discovery“ ausgeführt worden sein.
Die Progressbar zeigt dabei den Status an wie viele VMs gestartet sind.



8.2.6.2 Start Scenario

Nachdem einige oder alle (je nach Wunsch des Benutzers) VMs aufgestartet sind, kann das Szenario über den Knopf „Start Scenario“ gestartet werden. Die Progressbar aktualisiert sich automatisch und zeigt anschliessend den Fortschritt aller Tasks an:



8.2.6.3 Counters

Die Counters zeigen dem Benutzer die maximale Anzahl an VMs an. Nach der Auswahl eines Szenarios werden die übrigen Slots automatisch berechnet.

8.2.7 Current Scenario

Wenn ein Szenario gestartet wurde, wird hier der Name angezeigt.

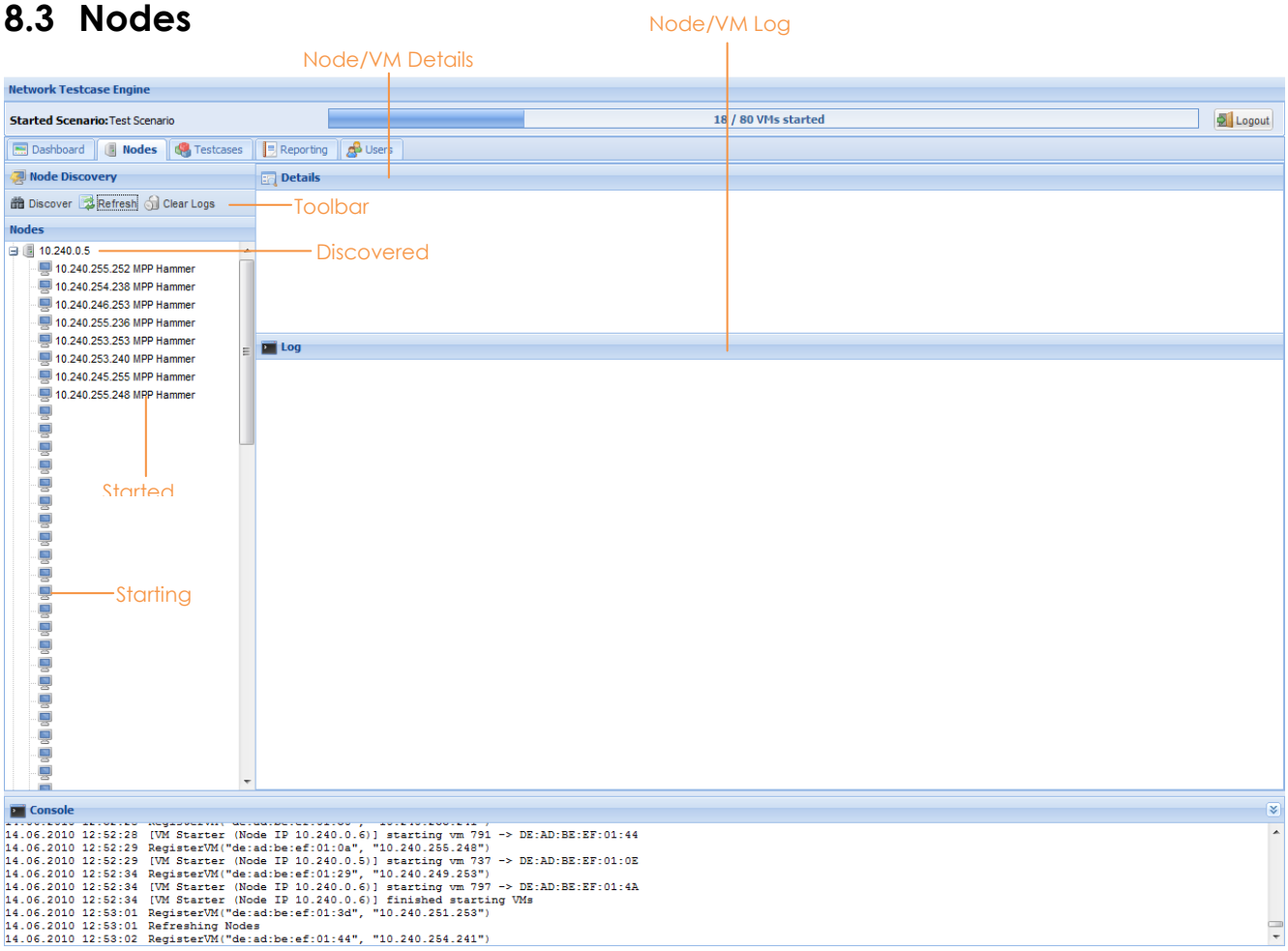
8.2.8 Logout

Der Benutzer kann sich vom System abmelden. Ein laufendes Szenario wird nicht abgebrochen. Der Benutzer kann sich später wieder anmelden, um die Resultate zu begutachten.

8.2.9 Console

In der Console werden die Interaktionen des Benutzers aufgenommen und Fehler der Applikation angezeigt. Die Console kann bei Wunsch ausgeblendet werden durch einen Mausklick auf den Doppelpfeil oben rechts. Die Grösse der Console kann mit der Maus verändert werden.
Wenn der Scrollbalken der Console ganz unten ist, dann scrollt sie automatisch bei neuen Einträgen nach unten.

8.3 Nodes



8.3.1 Toolbar

Die Toolbar bietet die Interaktionen für den Benutzer auf dieser Sicht.

8.3.1.1 Discover

Mittels „Discover“ werden die verfügbaren Nodes dem NTE-Service gemeldet und angezeigt. Falls ein Szenario schon am Laufen ist und eine Node Discovery ausgeführt wird, werden alle VMs auf den Nodes beendet.

8.3.1.2 Refresh

Mit „Refresh“ kann die Ansicht mit den Nodes und VMs aktualisiert werden um Änderungen ersichtlich zu machen.

8.3.1.3 Clear Logs

Löscht die vorhandenen Logs der Nodes und VMs.

8.3.2 Discovered Nodes

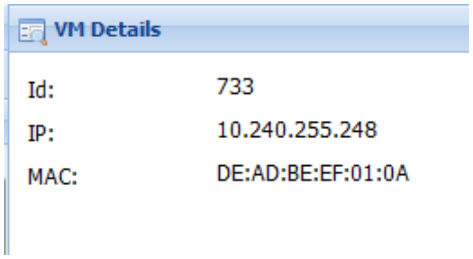
Nach dem Tätigen des Discovery Buttons werden hier die gefundenen Nodes angezeigt. Falls nichts angezeigt wird, „Refresh“ Button benutzen.

8.3.3 VMs

Nachdem in der Dashboardsicht der Button „Start VMs“ geklickt wurde, starten die VMs auf und werden unter der entsprechenden Node angezeigt.
VMs welche erfolgreich gestartet wurden, haben die IP und den Typ im Namen. Die VMs welche noch am starten sind werden angezeigt, haben jedoch weder IP noch Typ angezeigt (dieser wird erst nach dem Bootvorgang zugewiesen).

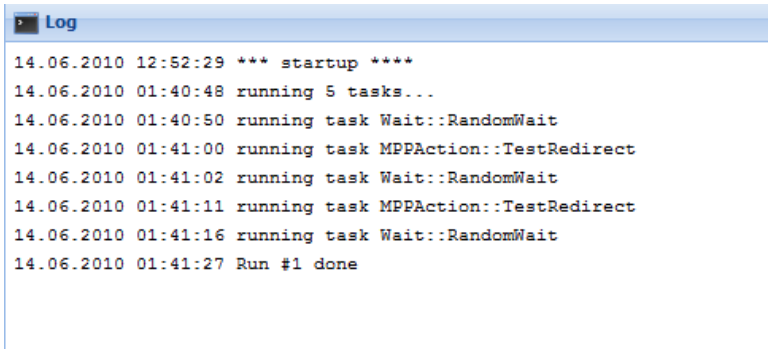
8.3.4 Node/VM Details

In diesem Bereich werden die Details, z.B. IP oder MAC, einer Node oder VM angezeigt. Um diese zu sehen muss eine eine Node oder VM angeklickt werden.

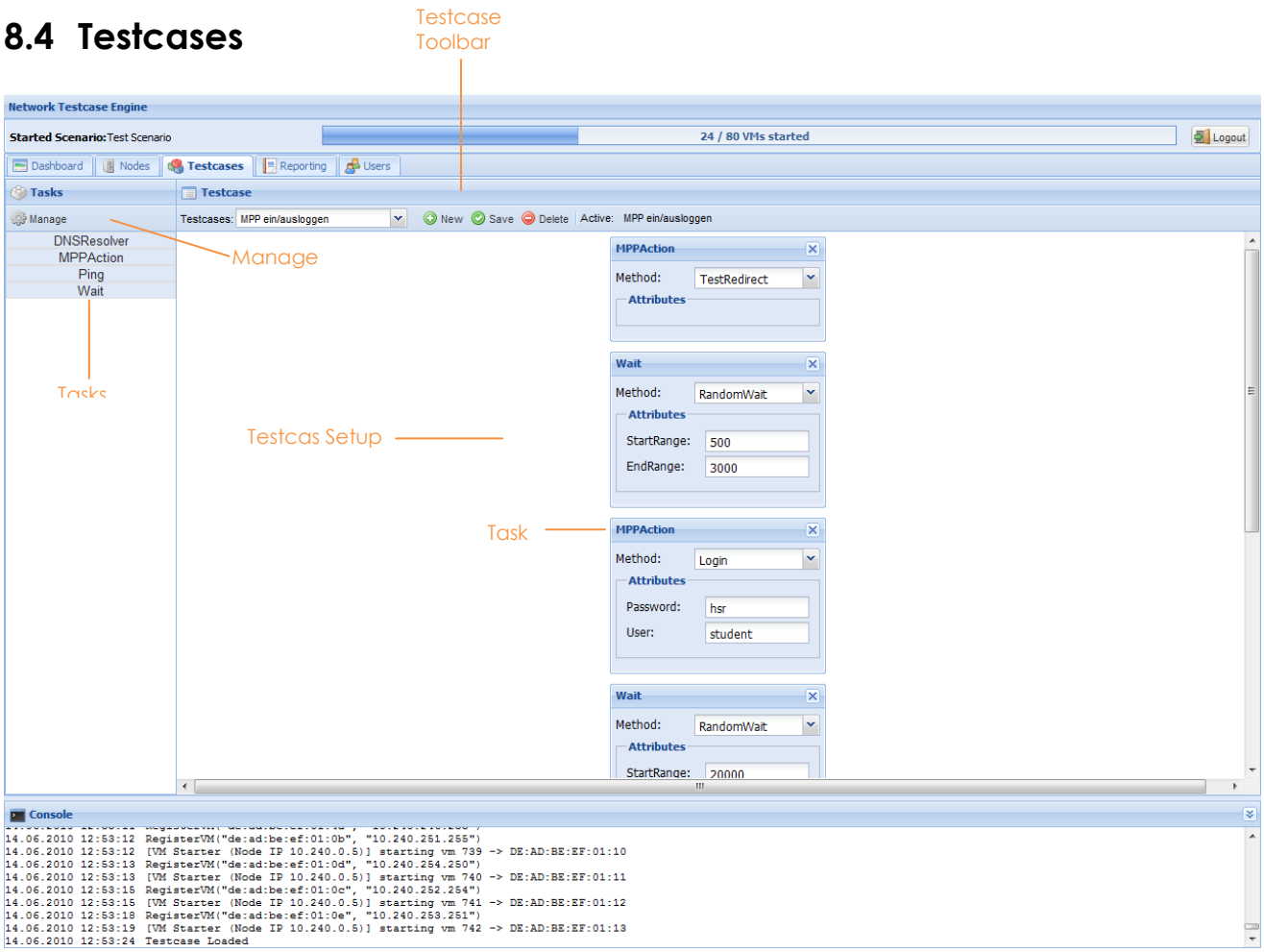


8.3.5 Node/VM Log

In diesem Bereicht werden die Logs zu einer Node oder VM Angezeigt. Um diese zu sehen muss eine Node oder VM angeklickt werden.

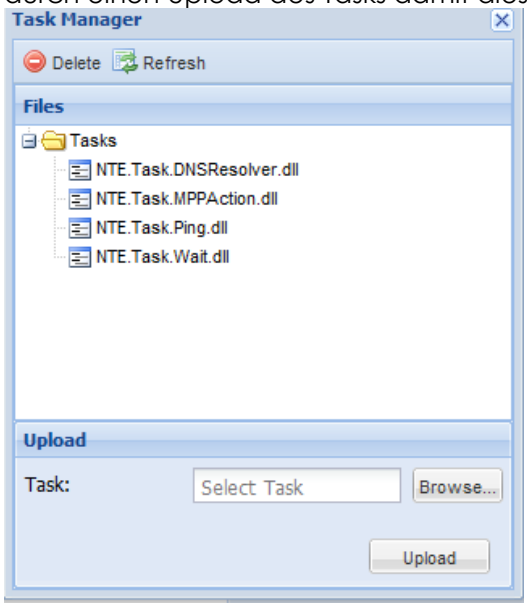


8.4 Testcases



8.4.1 Manage

Der Button "Manage" ermöglicht es dem Benutzer neue Task DLLs zu importieren. Dies geschieht durch einen Upload des Tasks damit dieser zur Verfügung steht.



8.4.2 Tasks

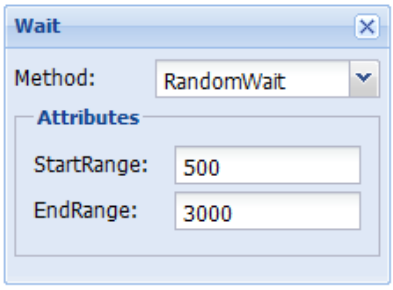
Die Tasks welche zur Verfügung stehen um einen Testcase zu erstellen.

8.4.3 Testcase Setup

In diesem bereich wird ein Testcase zusammengebaut. Ein neuer Task kann mittels Drag und Drop hinzugefügt werden. Um einen Task wieder zu entfernen, kann auf das X geklickt werdet.
Die Reihenfolge der Tasks im Testcase kann einfach durch Drag und Drop festgelegt werden. Dabei gillt, dass der Task zu oberst der Erste ist.

8.4.4 Task

Ein Task wird im Testcase Setup mit all seinen Eigenschaften dargestellt.



8.4.4.1 Combobox

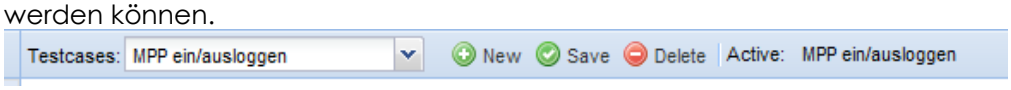
Hier wird die gwünschte Methode des Tasks gewählt, welche ausgeführt werden soll. Die Optionen zu den Einstellungen der Methode werden automatisch angezeigt.

8.4.4.2 Attribute

Die Optionen zur Taskmethode können hier nach wunsch eingestellt werden.

8.4.5 Testcase Toolbar

In der Testcase Toolbar befinden sich die Interaktionen welche von einem Benutzer durchgeführt werden können.



8.4.5.1 Testcase Combobox

In der Combobox werden alle vorhandenen Testcases aufgelistet. Um eines auszuwählen kann per Mausclick die Liste angezeigt werden oder der Name eingetippt werden. Nach dem Auswählen wird der Testcase automatisch geladen.

8.4.5.2 New

Beim drücken diese Buttons wir ein neuer Testcase initalisiert mit einem Zufälligem Namen. Dieser kann jedoch geändert werden durch Eingabe des gwünschten Namens in die Combobox.

8.4.5.3 Save

Dieser Button speichert alle änderungen welche gemacht wurden am Testcase. AlleÄnderungen werden erst übernommen nachdem dieser Aktion getätigt wurde. Falls dieser Button nicht gedürckt wird und ein Testcase gewechselt wird, gehen alle Änderungen verloren.

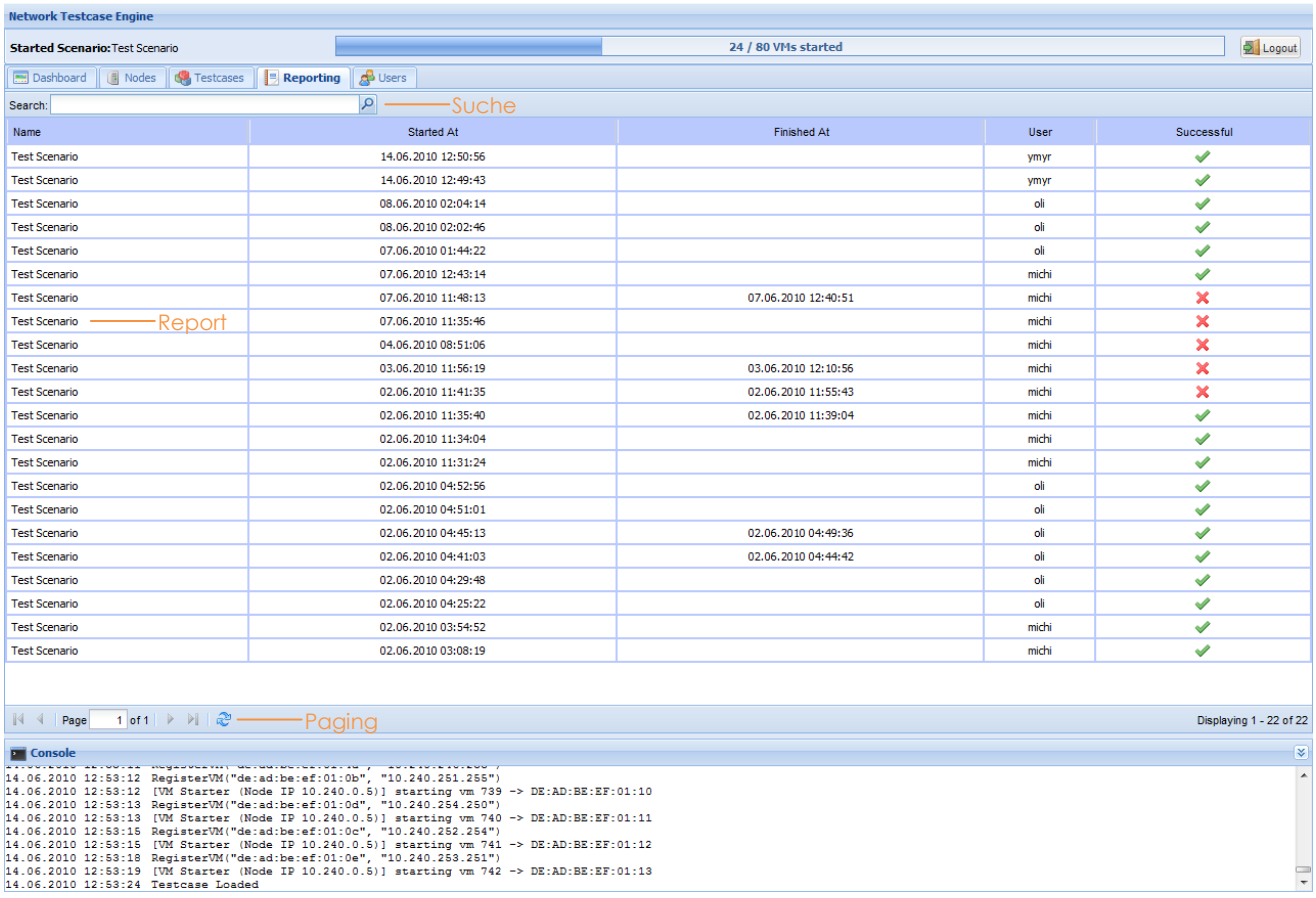
8.4.5.4 Delete

Dise Aktion löscht den ausgewählten Testcase. Beim löschen ist zu beachten, dass alles was mit dem Testcase in Verbindung steht auch gelöscht wird.

8.4.6 Active

Hier wird der Aktuell gwählte Testcase angezeigt, damit vom Benutzer gesehen kann auf welchem Testcase er momentan die Änderungen vornimmt und nicht ausversehen am falschen Testcase arbeitet und dies abspeichert.

8.5 Reporting

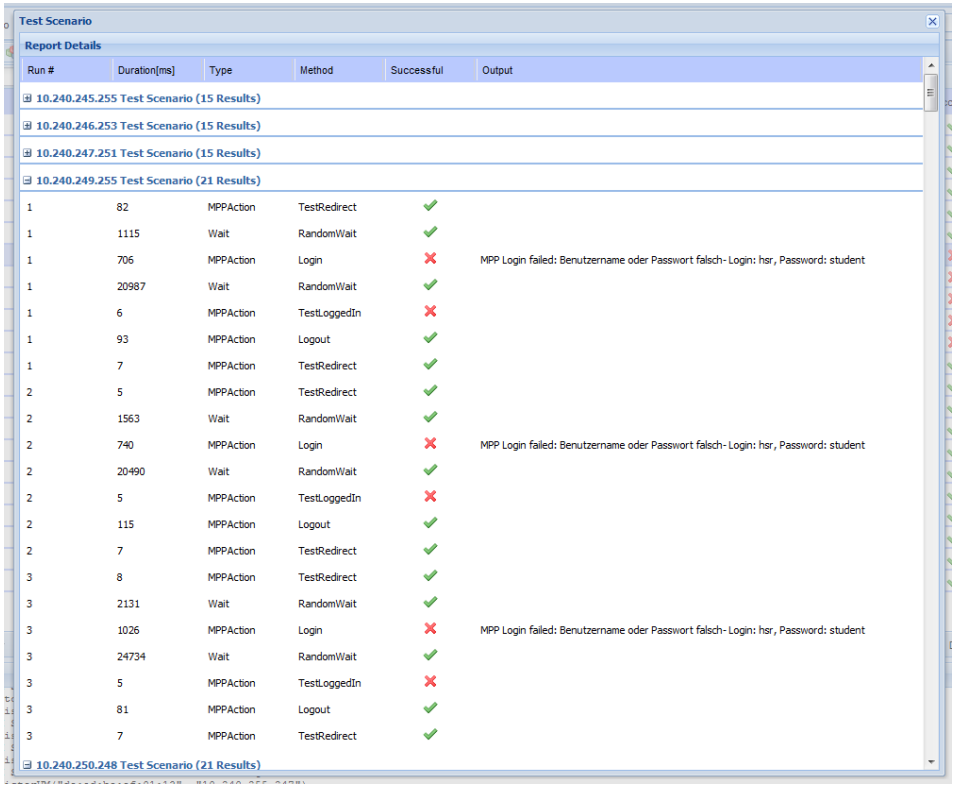


8.6 Suche

Der Benutzer kann nach spezifischen Reports suchen. Dazu dient dieses Suchfeld. Eingegeben kann irgend ein Text und nach diesem wird gesucht. Egal ob Datum oder Wort. Die Suche geht über alle Felder. Als einziges zu beachten ist die Suche nach Erfolgreichen Reports. Dazu muss True oder False eingegeben um nach diesem Kriterium erfolgreich zu Suchen.

8.6.1 Report

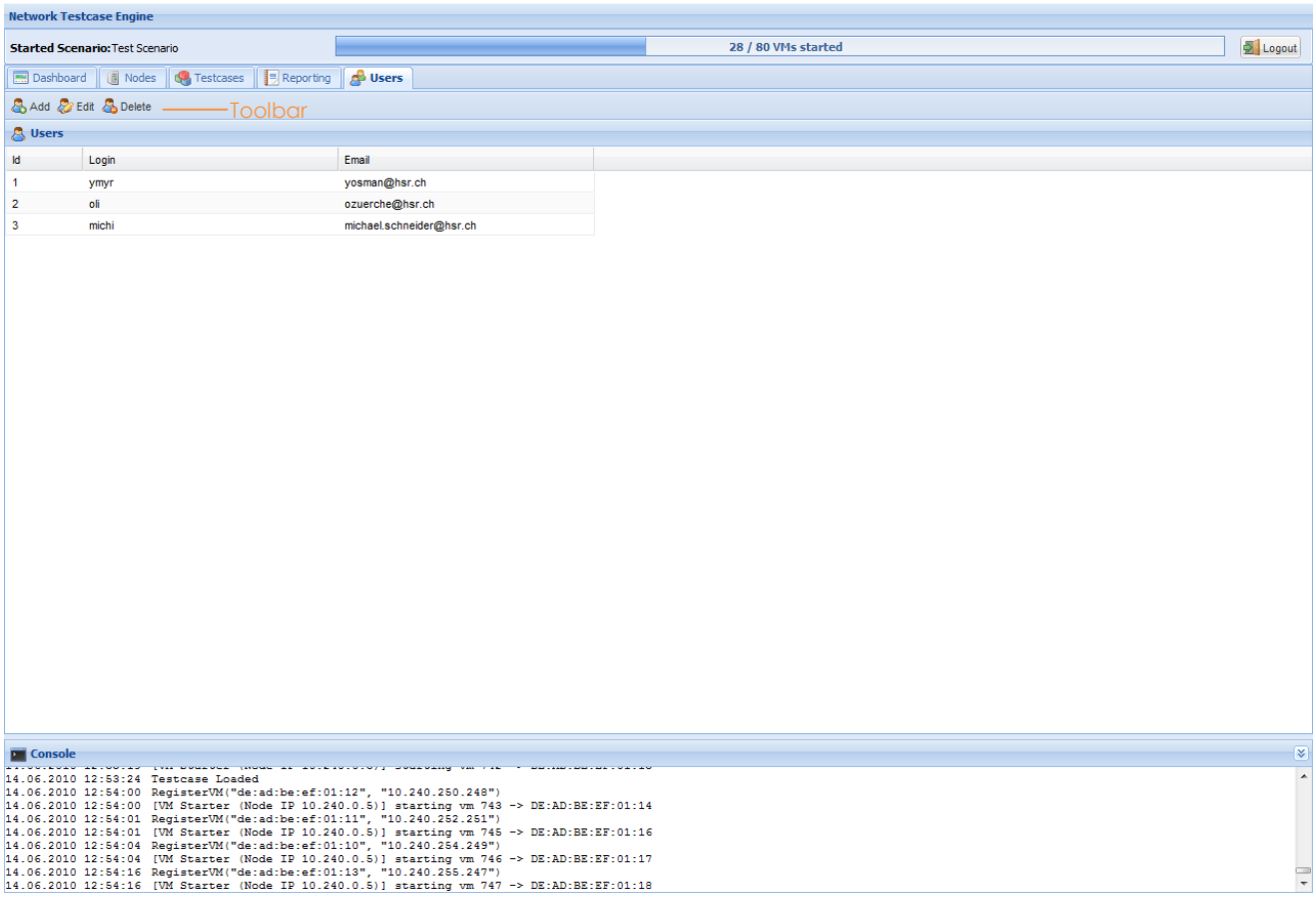
Die ausgeführten Szenarien werden tabellarisch angezeigt. Falls ein Szenarioverlauf genauer interessiert, kann ein Detailierter Report zum Szenario aufgerufen werden. Dies geschiet durch ein Doppelklick auf dem gewünschten Report.



8.6.2 Paging

Die aktuellsten Durchläufe werden immer zu oberst angezeigt und auf der ersten Seite werden die letzten 30 Durchläufe gezeigt. Falls ältere Reports Angeschaut werden müssen, kann mittels dem Paging zurück geblättert werden.

8.7 Users

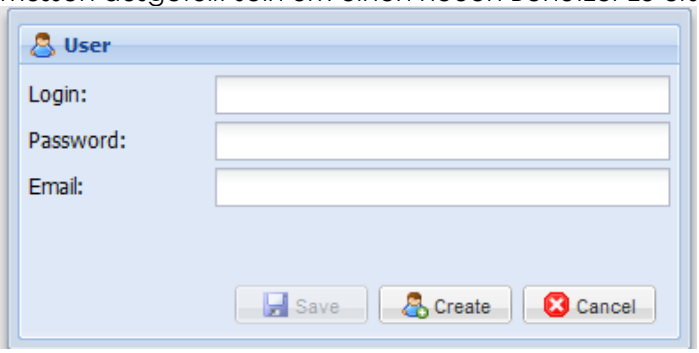


8.7.1 Toolbar

In der Toolbar befinden sich die Interaktionen welche von einem Benutzer durchgeführt werden können.

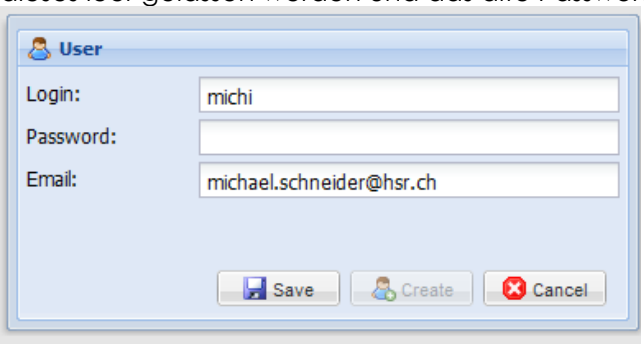
8.7.1.1 Add

Dieser Button fügt einen neuen Benutzer hinzu. Dabei öffnet sich eine Eingabemaske. Alle Felder müssen ausgefüllt sein um einen neuen Benutzer zu erstellen.



8.7.1.2 Edit

Um einen Benutzer zu Editiren muss dieser zuerst markiert werden. Danach kann der Button gedrückt werden und die Eingabemaske erscheint. Falls das Passwort nicht gewechselt werden soll, dann kann dieses leer gelassen werden und das alte Passwort bleibt bestehen.



8.7.1.3 Delete

Um den gewünschten Benutzer zu löschen, muss dieser Markiert werden. Danach kann der Button gedrückt werden und der Benutzer ist gelöscht.