INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Fortgeschrittenenpraktikum

Erstellung eines webbasierten Konverters für virtuelle Maschinen unter Verwendung von Webservices

Hans Moog

Draft vom 6. Juni 2011

INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Fortgeschrittenenpraktikum

Erstellung eines webbasierten Konverters für virtuelle Maschinen unter Verwendung von Webservices

Hans Moog

Aufgabensteller:	Prof. Dr. D. Kranzlmüller
Betreuer:	Nils Gentschen Felde Tobias Lindinger Johannes Watzl
Abgabetermin:	6. Juni 2011

Hiermit versichere ich, dass ich die vorliegende Projektarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 6. Juni 2011

.....

(Unterschrift des Kandidaten)

Abstract

Die einfache und kostengünstige Verwaltung von virtuellen Infrastrukturen hat dafür gesorgt, dass die Verbreitung von Virtualisierungslösungen in den letzten Jahren stark zugenommen hat. Zusammen mit der Bedeutung stieg aber auch die Anzahl der Anbieter und somit der Bedarf an einer Konvertierungsmöglichkeit zwischen den unterschiedlichen Formaten der Hersteller.

Im Rahmen dieser Arbeit wird zunächst eine Analyse vorgenommen, die bisherige Konverter bezüglich ihrer Tauglichkeit für eine Automatisierung untersucht. Anschließend wird mit diesen Erkenntnissen ein Konverter für virtuelle Maschinen implementiert, der es erlaubt, die unterschiedlichen proprietären Formate der einzelnen Hersteller von Virtualisierungslösungen ineinander zu überführen. Da die Anzahl der existierenden Lösungen viel zu groß ist, um tatsächlich alle Produkte zu unterstützen, wird primär ein Rahmenwerk zur Verfügung gestellt, welches in der Lage ist, die gewünschten Aufgaben durchzuführen. Die Funktionalität wird am Beispiel von drei verschiedenen Formaten gezeigt, die exemplarisch implementiert werden. Bei der Erstellung des Rahmenwerks wird darauf geachtet, dass ohne größeren Aufwand weitere Produkte hinzugefügt werden können.

Das Hauptaugenmerk liegt neben der Erweiterbarkeit auch auf dem Einsatz von Webservices als automatisierbare Schnittstelle. Aufbauend auf dieser Schnittstelle wird im Anschluss eine Webanwendung programmiert, welche als Machbarkeitsnachweis die Aufgabe übernimmt, die Leistungsfähigkeit und Funktionsweise der darunterliegenden Architektur zu zeigen.

Die Weboberfläche wird mit PHP und Ajax realisiert und läuft auf einem Apache Server mit Anbindung an eine MySQL-Datenbank. Da die eingesetzten Konvertierungsprogramme unterschiedliche Betriebssysteme benötigen, wird ein mehrere Rechner umfassender Verbund erstellt, der ebenfalls über Webservices kommuniziert und so sicherstellt, dass alle benötigten Umwandlungspfade abgedeckt werden können. Die Konvertierung wird jobweise ausgeführt und ermöglicht durch den Einsatz einer Queue auch das Einstellen und Planen mehrerer Konvertierungsaufgaben, sodass eine Batch-Fähigkeit gegeben ist.

Inhaltsverzeichnis

1. Einleitung

Virtualisierung zählt seit einigen Jahren zu den bestimmenden Themen der IT-Infrastruktur. Gerade im Umfeld großer Rechenzentren erreichte man durch die einfachere Verwaltung und die Möglichkeit, viele unabhängige Systeme auf wenigen Hardwareeinheiten zu konsolidieren, eine starke Energie- und Kosteneinsparung. Diese wirtschaftlichen Vorteile führten zu einer sehr schnellen und intensiven Verbreitung im genannten Umfeld. Das rasche Wachstum hatte aber nicht nur eine Bedeutungszunahme der Technologie, sondern auch eine erhebliche Diversifikation zur Folge. Während es Anfang des Jahrtausends nur wenige spezialisierte Anbieter von Virtualisierungslösungen gab, findet man heute eine Vielzahl breit aufgestellter Unternehmen, die unterschiedlichste Ansätze verfolgen und ein ganzes Spektrum an Plattformen und Produkten in diesem Bereich zur Verfügung stellen. Der daraus resultierende Wettbewerb zwischen den Protagonisten brachte leistungsfähige und kostengünstige Lösungen hervor, die diese Technologie auch für kleine und mittelständische Unternehmen interessant machen. Dies eröffnet weiteres Wachstumspotential.

1.1. Motivation

Der oben beschriebene starke Konkurrenzkampf hat neben den positiven Effekten aber auch negative Auswirkungen. So wird derzeit mit aller Macht versucht, sich Marktanteile zu sichern und durch protektionistische Maßnahmen die eigene Position zu festigen. Betrachtet man die Konvertierungsmöglichkeiten zwischen den proprietären Formaten der einzelnen Anbieter, wird dieser Umstand besonders deutlich. Es fällt auf, dass es kaum Möglichkeiten gibt, vom Format eines Anbieters in das eines anderen zu wechseln, ohne dabei größere Hindernisse überwinden zu müssen. Zum einen fehlen in den Programmen der Hersteller die Konvertierungspfade, und zum anderen wird oft eine Menge Hintergrundwissen vorausgesetzt. Dies macht es einem Anwender oder Administrator ohne tiefgreifende Kenntnisse über die spezifische Implementierung des Anbieters beinahe unmöglich, seine Virtualisierungsplattform frei zu wählen oder nachträglich zu wechseln.

Ein weiterer Hemmschuh ist die fehlende Automatisierbarkeit von Konvertierungsaufgaben, da es weder einheitliche Programmier-, noch geeignete Kommandozeilenschnittstellen gibt, die dies ermöglichen würden. Daraus resultiert ein wesentliches Problem mit einem neuen großen Trend – dem Cloud-Computing. Denn auf diese Art und Weise lassen sich Cloud-Lösungen für virtuelle Maschinen nur in homogenen Umgebungen realisieren und schließen einen Umzug von einer Cloud in die andere nahezu aus.

1.2. Aufgabenstellung

Um den eingangs genannten Problemen entgegenzuwirken, soll im Rahmen dieser Arbeit ein auf Webservices basierender Konverter für virtuelle Maschinen programmiert werden, der es erlaubt, die Formate der unterschiedlichen Anbieter möglichst verlustfrei ineinander

zu überführen. Damit die Lernkurve für einen potentiellen Anwender möglichst klein gehalten werden kann, soll der Konvertierungsvorgang zudem standardisiert werden, sodass auf anbieterspezifische Aspekte der Virtualisierung weitestgehend verzichtet werden kann. Exemplarisch soll darauf aufbauend ein Webinterface implementiert werden, welches die programmierte Programmierschnittstelle verwendet und somit ein Werkzeug bereitstellt, dass ohne weiteres Hintergrundwissen von jedem Anwender bedient werden kann.

1.3. Vorgehensweise

Da die Vielfalt der Firmen und Produkte im Bereich der Virtualisierung viel zu groß ist, um tatsächlich alle Fälle einer Konvertierung abzudecken, beschränkt sich diese Arbeit auf die Betrachtung einiger ausgewählter Beispiele. Die Arbeit ist somit als Machbarkeitsnachweis zu verstehen und versucht getroffene Annahmen so allgemein zu halten, dass sie für alle möglichen Anwendungsfälle ihre Gültigkeit behalten.

Um ein grundlegendes Verständnis für die Thematik zu schaffen, wird im Kapitel 2 zunächst auf einige theoretische Grundlagen eingegangen. Insbesondere der Aufbau einer virtuellen Maschine steht dabei im Mittelpunkt, da dieses Thema eine zentrale Rolle bei der Konvertierung spielt.
Anschließend wird aufbauend auf diesen Betrachtungen im Kapitel 3 der momentane Stand der Technik analysiert und auf Tauglichkeit hinsichtlich der geplanten Implementierung untersucht. Von besonderem Interesse sind hier vor allem die Herausforderungen, die im Zusammenhang mit der Aufgabenstellung stehen.
Im darauffolgenden Kapitel werden Lösungsansätze für die gefunden Herausforderungen diskutiert und die Anforderungen an die Implementierung verfeinert und ausformuliert.
In Kapitel 5 wird ein Systementwurf vorgestellt, der diesen Anforderungen gerecht wird. Hierbei wird vor allem auf die Architektur der zu programmierenden Software eingegangen. Kapitel 6 beschäftigt sich dann mit der Umsetzung und praktischen Lösung der angesprochenen Punkte. Im Anschluss wird das Webinterface vorgestellt und die Funktionsweise der Software erläutert.
Im letzten Kapitel wird ein Fazit aus den Ergebnissen dieser Arbeit gezogen und ein Ausblick auf mögliche darauf aufbauende Arbeiten gegeben.

2. Grundlagen

Im Folgenden wird auf grundlegende Begriffe und Zusammenhänge eingegangen, die es auch unvorbereiteten Lesern ermöglichen soll, den Ausführungen in dieser Arbeit zu folgen.

2.1. Virtualisierung und virtuelle Maschinen

Den Begriff der Virtualisierung allgemein zu definieren ist schwierig, da er eine Vielzahl unterschiedlicher Ansätze und Konzepte umfasst. Man kann zwar sagen, dass die Intention hinter allen Virtualisierungstechniken in der Regel die Aufteilung von vorhandenen Ressourcen beinhaltet, um diese neu zu verteilen und dadurch effizienter zu nutzen, doch bringt einem diese Aussage keinerlei Information darüber, wie diese Aufteilung technisch realisiert wird. Um den Begriff trotzdem definieren zu können, grenzen wir die Begriffsdefinition auf den für diese Arbeit relevanten Teil ein und betrachten nur die sogenannte Systemvirtualisierung. [ITW11]

Bei dieser Form der Virtualisierung geht es um die Bereitstellung einer virtuellen Maschine als gekapselte Entität, die ähnlich einem physisch vorhandenen Rechner betrieben werden kann. Das installierte Gastbetriebssystem benötigt dabei in der Regel keine Anpassungen. In der Tat ist es sogar so, dass das Gastbetriebssystem und auch andere Rechner im Netz nicht trivial einen Unterschied zu einem echten physischen Rechner feststellen können.

Da die Hardware des Rechners nicht wirklich vorhanden, sondern nur von einer Software (dem sogennannten Hypervisor) bereitgestellt wird, können die vorhanden Ressourcen nahezu beliebig verteilt werden, was eine schnelle Reaktion auf Änderungen in den Systemanforderungen einzelner Rechner und Anwendungen ermöglicht. Weitere Vorteile sind:

- Die Möglichkeit, mehrere teilweise sogar unterschiedliche Gastsysteme auf einem Host zu betreiben, führt zu einer besseren Energie- und Hardwareauslastung.
- Da wichtige Aufgaben wie Neustarten, Herunterfahren, Bereitstellen usw. über ein Verwaltungsprogramm ohne physische Interaktion vor Ort durchgeführt werden können, wird die Verwaltung vereinfacht.

2.2. Aufbau einer virtuellen Maschine

Betrachtet man den Aufbau einer virtuellen Maschine im Detail, so stellt man fest, dass virtuelle Maschinen oft in einem Container ausgeliefert werden, der aus zwei wesentlichen Teilen besteht: Disk-Images und Hardwarespezifikation (Abbildung 2.1).

2.2.1. Disk-Images

Die Disk-Images beinhalten die Daten der Festplatten, die in der virtuellen Maschine „verbaut“ sind. Oft liegen die Abbilder in Form einzelner Dateien vor, die die Rohdaten blockweise strukturiert – ähnlich einer physischen Festplatte – abbilden. Es gibt momentan keine

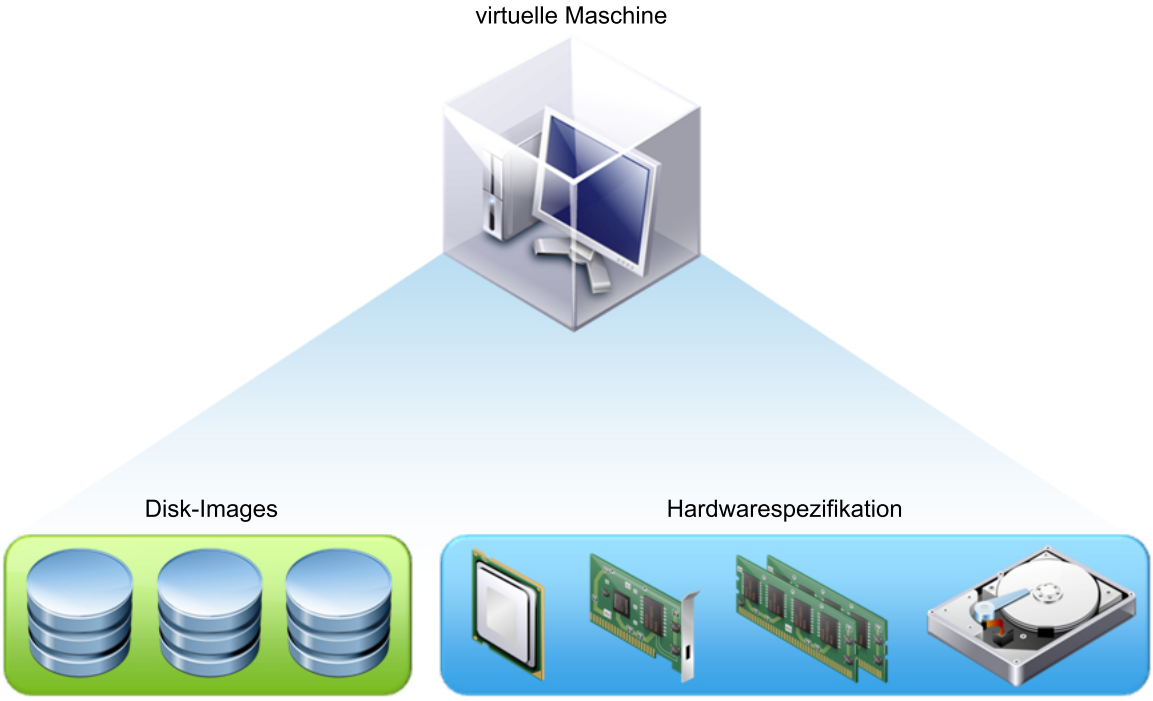

Abbildung 2.1.: Schematischer Aufbau einer virtuellen Maschine (Icons aus [Ste10])

Standardisierung, die festlegt, wie ein Disk-Image aufgebaut sein soll. Daher haben viele Anbieter ihre eigenen proprietären Formate entwickelt, die nicht zwangsläufig nur für die Virtualisierung eingesetzt werden. Die bekanntesten Formate werden im folgenden Abschnitt kurz beleuchtet.

VMDK (Virtual Machine Disk)

Das VMDK-Format wurde von VMware entwickelt und dient als Basis für nahezu alle VMware Produkte. Es wird aber auch von anderen Anbietern (QEMU, VirtualBox, Sun xVM, SUSE Studio) unterstützt und verwendet. Das Format wurde für das kontinuierliche Übertragen der Daten über das Netzwerk (Streaming) optimiert und eignet sich daher besonders für den Einsatz auf einem zentralen Netzwerkspeicher (NAS/SAN). So lassen sich beispielsweise festplattenlose Server realisieren, die in einem Hochverfügbarkeitsverbund zusammengeschaltet sind und auf gemeinsame virtuellen Festplatten zugreifen. Im Falle eines Ausfalls kann dann ein anderer Server des Verbunds nahtlos die Arbeit des defekten Systems übernehmen, ohne dabei erst die Festplatteninhalte herunterzuladen zu müssen. [VMw07]

VHD (Virtual Hard Disk)

Das Format mit dem Namen VHD wurde von Connectix erfunden und später von Microsoft aufgekauft, um es für die eigenen Virtualisierungslösungen (Hyper-V, Microsoft Virtual PC) zu verwenden. Seit Windows 7 werden VHD-Abbilder auch vom nativen Windows-Bootmanager unterstützt, was es ermöglicht, mehrere Betriebssysteme auf einem System zu installieren und auszuführen, ohne dabei einen Hypervisor einsetzen zu müssen. VHD-

Festplatten unterstützten wie die meisten anderen virtuellen Formate eine Snapshot-Funktionalität und erlauben so auch ohne Hypervisor den Aufbau von Testumgebungen, da Änderungen jederzeit wieder rückgängig gemacht werden können. [Mic10a]

VDI (Virtual Desktop Image)

Standardmäßig benutzt der Virtualisierer VirtualBox von Oracle das eigens für dieses System entwickelte VDI-Format. Neben den gängigen Funktionen wie Snapshots und dynamisch wachsenden Abbildern erlaubt das genannte Format die Limitierung der Festplattengeschwindigkeit. Nennenswert ist auch die Tatsache, dass VDI zwar immer noch das Standard-Format für VirtualBox-Container darstellt, der Trend aber zur Benutzung von anderen Formaten (wie VMDK oder VHD) geht, die inzwischen ebenfalls vollständig unterstützt werden. [Ora11]

qcow/qcow2

Das vom Virtual Machine Monitor QEMU benutzte Format QCOW bzw. QCOW2 (die neuere Version) besitzt im Gegensatz zu den bisher betrachteten Formaten eine grundsätzlich andere Funktionalität. Es basiert auf der sogenannten „Copy on Write“-Methode, die es ermöglicht, virtuelle Festplatten basierend auf einem Basisabbild bereitzustellen, das als Vorlage für alle darauf aufbauenden virtuellen Festplatten dient. Statt Änderungen direkt auf die Basisfestplatte zu schreiben, werden die geänderten Blöcke an eine neue Position geschrieben, ohne die alten Daten anzutasten. Anschließend wird der Zeiger auf den geänderten Datenblock umgelenkt und der Festplatteninhalt gültig, ohne die alten Daten zu überschreiben. Diese können somit immer noch als Vorlage für weitere Festplatten dienen und erlauben auf diese Weise eine einfache Möglichkeit, schnell und unkompliziert vorgefertigte Systemabbilder zu verteilen. Die ursprünglichen Daten dienen dabei zusätzlich als Snapshots, die aber beim Anlegen keine zusätzliche Zeit in Anspruch nehmen. Diese Snapshots erlauben ein schnelles Zurücksetzen bei Fehlern oder ungewollten Änderungen. [QEM08]

2.2.2. Hardwarespezifikation

Unter der Hardwarespezifikation versteht man eine Liste der im Rechner verbauten Hardwarekomponenten. Zusätzlich zur bloßen Nennung der Komponenten wird auch deren Beziehung zueinander angegeben. So wird beispielsweise festgelegt, an welchem Anschluss des SCSI-Controllers die Festplatte angeschlossen oder in welchem PCI-Slot die Grafikkarte verbaut ist. Auf diese Art und Weise entsteht eine exakte Beschreibung der Hardware, die es dem Hypervisor erlaubt, die virtuelle Maschine zu betreiben. Analog zu den Disk-Images haben die Hersteller auch für die Beschreibung der Hardware eigene Formate entwickelt, die meist auf dem XML-Format basieren und keinen eigenen „Namen“ haben.

Open Virtualization Format (OVF)

Das im Gegensatz zu den angesprochenen Formaten anbieterunabhängige OVF der Distributed Management Task Force bildet die Hardwarespezifikation ebenfalls in einer XML-Datei ab. Bei der Entwicklung des Open Virtualization Formats wurde darauf abgezielt, einen Standard für die Angabe der virtuellen Hardware zu schaffen. Dementsprechend spielten

vor allem die folgenden Aspekte eine wichtige Rolle bei der Ausarbeitung der Spezifikation: [DMT10]

- Optimierte Verteilung: Das Format unterstützt Lizenzen, Integritätsprüfungen und eine Inhaltsverifikation des Pakets.
- Einfaches und automatisiertes Bedienerlebnis: Benutzerlesbare Metadaten und Paketinformationen werden in die Beschreibung eingebettet. Dadurch ist ein automatisierte Gültigkeitsprüfung möglich.
- Unterstützung für Einzel- und Multi-VM-Konfigurationen: In einer OVF-Datei können auch komplexe virtuelle Umgebungen abgebildet werden, die aus mehreren virtuellen Maschinen bestehen.
- Portables Paketformat: Das OVF ist grundsätzlich plattformunabhängig in Bezug auf den eingesetzten Hypervisor, erlaubt aber auch das Erfassen von plattformspezifischen Informationen. Alle bekannten Disk-Image-Formate werden unterstützt.
- Anbieter- und Plattformunabhängigkeit: Zusätzlich zur bereits genannten Unabhängigkeit hinsichtlich des eingesetzten Virtualisierers ist das Format auch unabhängig in Bezug auf das Gastsystem und die Hostplattform.
- Erweiterbarkeit: Das OVF-Schema ist so allgemein gehalten, dass auch zukünftige Anforderungen der Virtualisierungsindustrie ohne Änderungen am Standard abgebildet werden können. Auch das Einbetten von plattformspezifischen Zusatzinformationen ist möglich.
- Mehrsprachigkeit: Die Beschreibungen des Pakets können in unterschiedlichen Sprachen verfasst werden, um möglichst viele Marktteilnehmer zu erreichen.
- Offener Standard: Das OVF ist durch die Zusammenarbeit mehrerer großer Anbieter entstanden und zielt darauf ab, in Zukunft den Industriestandard für portable virtuelle Maschinen bereitzustellen.

Die angesprochenen Punkte sind grundsätzlich sinnvoll und versprechen einen einfacheren Umgang mit virtuellen Maschinen. Im Laufe dieser Arbeit wird sich aber herausstellen, dass einige dieser Punkte zu Problemen mit derzeit existierenden Konvertern führen, da die Spezifikation zu allgemein gehalten ist und somit zu viel Spielraum für Interpretationen lässt. So ist beispielsweise noch nicht einmal festgelegt, in welcher „Einheit“ die zugeteilten Ressourcen angegeben werden müssen, weshalb viele Konverter nicht ohne weiteres mit den Dateien eines anderen Anbieters umgehen können. [DMT10]

3. Stand der Technik

In diesem Kapitel geht es darum, festzustellen, welche Konverter für den Einsatz im geplanten System geeignet sind und welche Herausforderungen sich für das neue System durch eventuell vorhandene Einschränkungen in der Funktionsweise der bereits existierenden Programme ergeben.

Um einen Eindruck vom momentanen Stand der Technik zu bekommen, wird deshalb zunächst ein Überblick über die derzeit existierenden Konverter gegeben. Im Mittelpunkt der Betrachtung stehen dabei vor allem die unterstützten Konvertierungspfade, die für die Aufgabenstellung, einen automatisierbaren Universalkonverter zu programmieren, von zentraler Bedeutung sind. Exemplarisch werden die drei größten Anbieter und deren Werkzeuge verglichen:

- VMware: VMware vCenter Converter
- Citrix: Citrix XenConvert
- Microsoft: System Center Virtual Machine Manager

Die daraus resultierenden Daten sind mehrdimensional und lassen sich nur sehr schwer in einer gemeinsamen Übersicht darstellen (Abbildung 3.1). Daher werden die Daten zur einfacheren Darstellung, entlang der Achse der dritten Dimension (Gastbetriebssystem) geschnitten und einzeln tabellarisch aufgelistet.

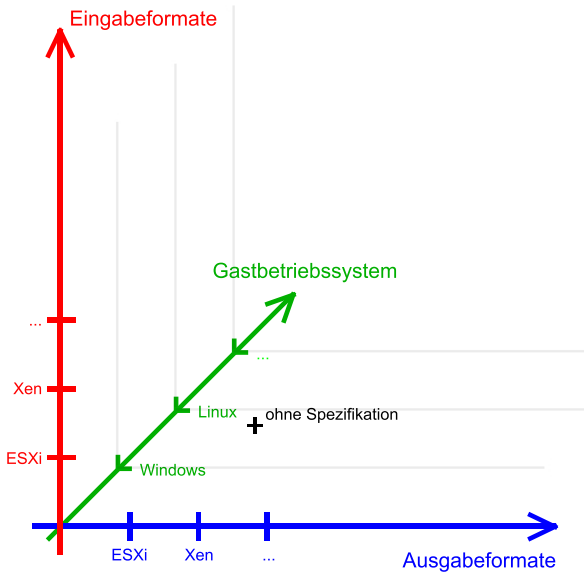

Abbildung 3.1.: Hohe Dimensionalität der Daten

3.1. VMware vCenter Converter

Das Konvertierungswerkzeug aus dem Hause VMware ist ein wizard-gesteuertes Programm, welches es ermöglicht, Maschinen aus den unterschiedlichsten Formaten und Quellen in eine VMware-Umgebung zu importieren. Es gibt je eine Version für windows- und linuxbasierte Systeme, die aber beide keine Kommandozeilenfähigkeit besitzen und somit nur schwer automatisierbar sind. Trotz seines Namens setzt das Programm keinen vCenter-Server (Name eines Verwaltungssystems von VMware, das es erlaubt große virtuelle Umgebungen zentral zu steuern) voraus. Es existiert vielmehr eine zusätzliche unabhängige Ausführung, die sich hinsichtlich der Funktionalität nur geringfügig von der Servervariante unterscheidet. So ist das P2V (Umwandlung von physischen in virtuelle Maschinen) für Linux der unabhängigen Variante vorbehalten, während die in vCenter integrierte Version exklusiv das Kopieren von ausgeschalteten virtuellen Maschinen und die zeitgesteuerte Konvertierung beherrscht. [VMw11]

3.1.1. Konvertierungspfade

In den Tabellen 3.1 und 3.2 werden die für die getroffene Anbieterauswahl relevanten Konvertierungsmöglichkeiten dargestellt. Wenn das Ziel der Konvertierung ein VMware-Format ist, bietet der Konverter relativ viele Pfade an, die zwar teilweise Einschränkungen hinsichtlich des Gastbetriebssystems machen, aber dafür die komplette Spezifikation übernehmen können. Bei der Betrachtung der Konvertierungsmöglichkeiten in ein anbieterfremdes Format fällt hingegen auf, dass keines der betrachteten Produkte unterstützt wird.

Eingabe / Ausgabe	ESX(i)	VMware Server	VMware Workst.	Xen	Hyper-V	Virtual PC
ESX(i)	inkl. Spezifik.	inkl. Spezifik.	inkl. Spezifik.	-	inkl. Spezifik.	inkl. Spezifik.
VMware Server	inkl. Spezifik.	inkl. Spezifik.	inkl. Spezifik.	-	inkl. Spezifik.	inkl. Spezifik.
VMware Workst.	inkl. Spezifik.	inkl. Spezifik.	inkl. Spezifik.	-	inkl. Spezifik.	inkl. Spezifik.
Xen	-	-	-	-	-	-
Hyper-V	-	-	-	-	-	-
Virtual PC	-	-	-	-	-	-

Tabelle 3.1.: VMware vCenter Konverter: Konvertierung von Windowssystemen

3.1.2. Fazit

Der VMware vCenter Converter ist ein relativ umfangreiches Werkzeug, welches viele wichtige Quellformate unterstützt. Trotzdem lassen sich selbst hier keine Maschinen von Citrix-basierten Systemen importieren. Leider ist die Konvertierung zudem eine Einbahnstraße (siehe Abbildung 3.2), da keine nennenswerten Exportmöglichkeiten in Fremdformate existieren.

Eingabe / Ausgabe	ESX(i)	VMware Server	VMware Workst.	Xen	Hyper-V	Virtual PC
ESX(i)	inkl. Spezifik.	inkl. Spezifik.	inkl. Spezifik.	-	-	inkl. Spezifik.
VMware Server	inkl. Spezifik.	inkl. Spezifik.	inkl. Spezifik.	-	-	inkl. Spezifik.
VMware Workst.	inkl. Spezifik.	inkl. Spezifik.	inkl. Spezifik.	-	-	inkl. Spezifik.
Xen	-	-	-	-	-	-
Hyper-V	-	-	-	-	-	-
Virtual PC	-	-	-	-	-	-

Tabelle 3.2.: VMware vCenter Konverter: Konvertierung von Linuxsystemen

Die fehlende Automatisierbarkeit ist ein weiterer Minuspunkt und disqualifiziert das Tool für unsere Aufgabenstellung.

3.2. Citrix XenConvert

Das Umwandlungswerkzeug von Citrix ist analog zum VMware Converter auch über einen grafischen Wizard bedienbar. Im Gegensatz zum genannten Produkt bietet es aber zusätzlich eine Kommandozeilenschnittstelle an, die es erlaubt, das Programm automatisiert aufzurufen. Es existiert derzeit keine Version für linuxbasierte Systeme, sodass man hinsichtlich des Betriebssystems eingeschränkt ist. Was bei diesem Programm auffällt, ist, dass das Zielformat abhängig vom Quellformat ist. Das heißt, dass man hier nicht einfach ein beliebiges Ursprungsformat wählen kann, um es in das gewünschte Zielformat zu überführen, sondern sich durchaus Gedanken machen muss, in welchem Format man dem Programm seine Daten präsentiert. [Cit10]

3.2.1. Konvertierungspfade

Die in den Tabellen 3.3 und 3.4 aufgelisteten Konvertierungspfade zeigen, dass der Pfad hin zu einem XenServer komplett abgedeckt wird. Zwar gehen in der Regel die Spezifikationen verloren, sodass diese beim Umzug manuell erstellt werden müssen, jedoch besteht zumindest die Möglichkeit mit dem genannten Programm alle betrachteten Produkte in diese Richtung umzuwandeln. Es gibt zwar – anders als beim VMware Converter – auch die Möglichkeit, virtuelle Maschinen in das Format eines anderen Herstellers zu exportieren, aber die hier betrachteten Anbieter sind nicht vertreten. Zudem ist zu sagen, dass im Gegensatz zum VMware vCenter Converter für den Export ein laufender XenServer (linuxbasiert) benötigt wird, und eine „Offline-Konvertierung“ somit nicht möglich ist.

Eingabe / Ausgabe	ESX(i)	VMware Server	VMware Workst.	Xen	Hyper-V	Virtual PC
ESX(i)	-	-	-	-	-	-
VMware Server	-	-	-	-	-	-
VMware Workst.	-	-	-	-	-	-
Xen	ohne Spezifik.	ohne Spezifik.	ohne Spezifik.	inkl. Spezifik.	ohne Spezifik.	ohne Spezifik.
Hyper-V	-	-	-	-	-	-
Virtual PC	-	-	-	-	-	-

Tabelle 3.3.: Citrix XenConvert: Konvertierung von Windowssystemen

Eingabe / Ausgabe	ESX(i)	VMware Server	VMware Workst.	Xen	Hyper-V	Virtual PC
ESX(i)	-	-	-	-	-	-
VMware Server	-	-	-	-	-	-
VMware Workst.	-	-	-	-	-	-
Xen	ohne Spezifik.	ohne Spezifik.	ohne Spezifik.	inkl. Spezifik.	ohne Spezifik.	ohne Spezifik.
Hyper-V	-	-	-	-	-	-
Virtual PC	-	-	-	-	-	-

Tabelle 3.4.: Citrix XenConvert: Konvertierung von Linuxsystemen

3.2.2. Fazit

Citrix XenConvert ist ein robustes Werkzeug, welches die wichtigsten Eingabeformate unterstützt. Allerdings ist die Konvertierung erneut nur einseitig, da keines der betrachteten Produkte als Ausgabeformat abgedeckt wird. Die integrierte Automatisierbarkeit prädestiniert es für den Einsatz im geplanten System. Als Schwierigkeit ist hier maximal der Fakt anzusehen, dass für eine Konvertierung zwei unterschiedliche Betriebssysteme benötigt werden, da das Tool nur in Zusammenarbeit mit einem XenServer seinen Dienst verrichtet.

3.3. System Center Virtual Machine Manager

Der System Center Virtual Machine Manager von Microsoft ist eigentlich gar kein reines Konvertierungstool, sondern die Management-Suite von Microsoft für den Hyper-V. Nichtsdestotrotz bietet sie im Verhältnis zu den anderen Konvertierungs- und Verwaltungsprogrammen von Microsoft die umfangreichsten Umwandlungsmöglichkeiten. Wie es der Name des Herstellers bereits vermuten lässt, gibt es den Manager (ähnlich wie den Citrix XenConvert) nur für Windows. Alle durchführbaren Aufgaben sind über eine Shell steuer- und somit auch leicht automatisierbar. Das Programm ist im Gegensatz zu den anderen beiden Kandidaten nicht kostenlos und benötigt eine zusätzliche Lizenz. Außerdem hat es relativ hohe Hardwareanforderungen, da das System auf dem Windows Server 2008 basiert. [Mic10b]

3.3.1. Konvertierungspfade

Der Konverter unterstützt zwar alle betrachteten Konkurrenzprodukte als Eingabeformat, doch macht er dabei entweder Einschränkungen hinsichtlich des Gastbetriebssystems oder verwirft die Spezifikation (Tabelle 3.5 und 3.6). Desweiteren gibt es auch hier keine Möglichkeit, virtuelle Maschinen in das Format eines anderen Herstellers umzuwandeln. Noch nicht einmal die Umwandlung in das Microsoft-eigene Format von Virtual PC wird unterstützt. Damit ist Microsoft der einzige der betrachteten Anbieter, der die eigene Produktpalette nicht zu 100% abdeckt.

3.3.2. Fazit

Der System Center Virtual Machine Manager ist ein leicht und komfortabel zu automatisierender Konverter, der allerdings sehr wenige Konvertierungspfade besitzt und große Einschränkungen hinsichtlich des Gastbetriebssystems macht. Gepaart mit der Tatsache, dass die Konvertierung auch hier nur in eine Richtung funktioniert, ist das Tool für die gesetzte Aufgabenstellung nur bedingt geeignet.

3.4. Zusammenfassung

Die einzelnen Aussagen über die möglichen Konvertierungspfade sind in Abbildung 3.2 zusammengefasst.

Wie in der Abbildung zu sehen ist, gibt es nur unzureichend viele Konvertierungspfade, die zudem noch stark verlustbehaftet sind, da häufig die Hardwarespezifikation komplett verloren geht. Wenn man nun noch davon ausgeht, dass man vom Ursprungs- ins Zielformat nur

Eingabe / Ausgabe	ESX(i)	VMware Server	VMware Workst.	Xen	Hyper-V	Virtual PC
ESX(i)	-	-	-	-	-	-
VMware Server	-	-	-	-	-	-
VMware Workst.	-	-	-	-	-	-
Xen	-	-	-	-	-	-
Hyper-V	inkl. Spezifik.	inkl. Spezifik.	inkl. Spezifik.	ohne Spezifik.	inkl. Spezifik.	inkl. Spezifik.
Virtual PC	-	-	-	-	-	-

Tabelle 3.5.: System Center Virtual Machine Manager: Konvertierung von Windowssystemen

Eingabe / Ausgabe	ESX(i)	VMware Server	VMware Workst.	Xen	Hyper-V	Virtual PC
ESX(i)	-	-	-	-	-	-
VMware Server	-	-	-	-	-	-
VMware Workst.	-	-	-	-	-	-
Xen	-	-	-	-	-	-
Hyper-V	-	-	-	ohne Spezifik.	inkl. Spezifik.	inkl. Spezifik.
Virtual PC	-	-	-	-	-	-

Tabelle 3.6.: System Center Virtual Machine Manager: Konvertierung von Linuxsystemen

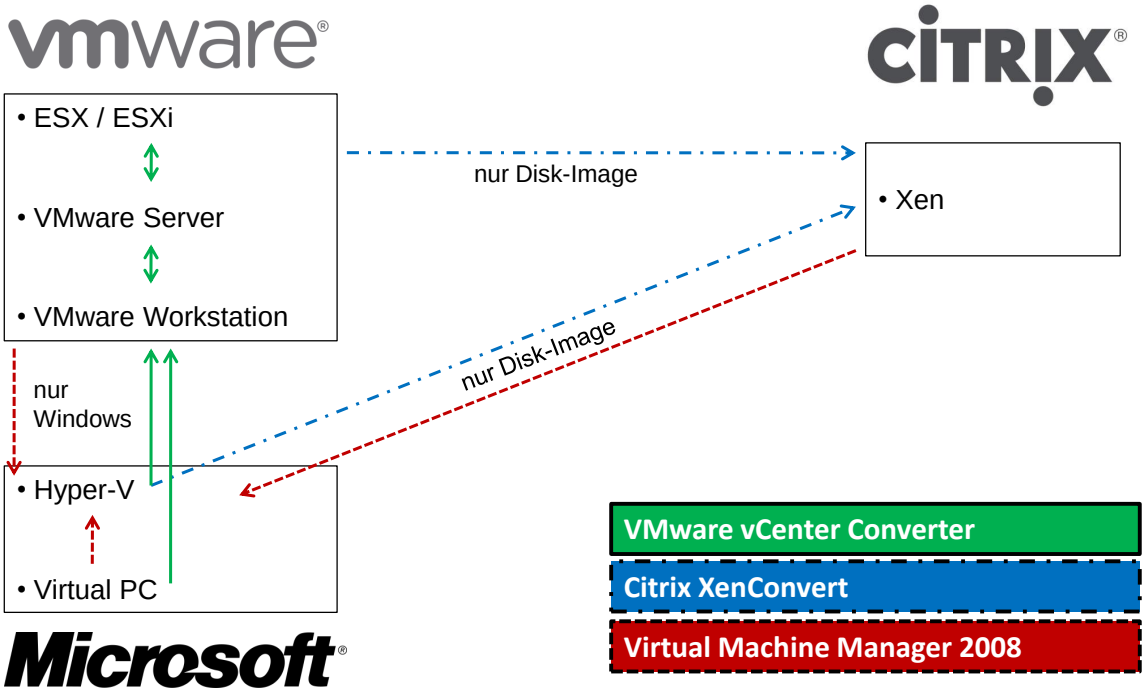

Abbildung 3.2.: Konvertierungspfade zwischen den Produkten der betrachteten Anbieter

über Zwischenschritte gelangt (z. B. VMware → Citrix), wird klar, dass der Konvertierungsvorgang mehr dem Spiel „Stille Post“, als einem zuverlässigen deterministischen Prozess, der als Basis für einen Universalkonverter dienen könnte, gleicht. Zusätzlich zu dieser Hauptproblematik gibt es noch eine Handvoll anderer Probleme:

- Die Konvertierung mit einem Tool ist meist nur in eine Richtung möglich.
- Oft geht die Spezifikation der Maschine verloren.
- Eine ausreichende Automatisierbarkeit ist nicht immer gewährleistet.
- Die Werkzeuge benötigen unterschiedliche Betriebssysteme (Linux ↔ Windows).
- Es gibt häufig Einschränkungen hinsichtlich des Gastbetriebssystems, was für einen nicht-invasiven Konverter ungeeignet ist, da er nicht ohne weiteres feststellen kann, welches Gastsystem installiert ist.
- Man benötigt viele Konverter, in die man sich meist erst einarbeiten muss, um von der Quelle zum Ziel zu kommen; daraus folgt eine hohe Lernkurve.

Es wird deutlich, dass die Aufgabenstellung relativ viele Anforderungen bereithält, die im Laufe der nächsten Kapitel gelöst werden müssen.

4. Anforderungen und Herausforderungen

Aus dem vorangegangenen Kapitel ergeben sich eine Menge Herausforderungen, die in diesem Abschnitt gelöst werden sollen. Auch für die im Aufgabentext genannten Anforderungen wie Automatisierbarkeit und Standardisierung werden Lösungsideen zusammengetragen, sodass im Anschluss mit der Planung und der darauf folgenden Implementierung begonnen werden kann.

4.1. Automatisierbarkeit

Die Automatisierbarkeit ist eine der zentralen Forderungen der Aufgabenstellung und wird daher zuerst betrachtet.

4.1.1. Anforderung

Der programmierte Konverter muss in der Lage sein, die Konvertierungsaufgaben ohne Zutun eines menschlichen Benutzers durchzuführen. Das heißt zunächst einmal, dass alle für die Konvertierung benötigten Informationen am Anfang übergebbar sein müssen, sodass das System während der Umwandlung keine weiteren Informationen abfragen muss (Batch-Fähigkeit). Zum anderen bedeutet es, dass der Konverter auch von fremden Softwaremodulen bedienbar sein muss, damit eine Integration in jedes beliebige Softwareprodukt einfach und schnell möglich ist (Bereitstellen einer Programmierschnittstelle).

4.1.2. Lösungsansatz

Durch eine Webservice-Schnittstelle (SOAP [Wor07]) lassen sich beide Aspekte relativ einfach abdecken. Da Webservices über Methodenaufrufe angesprochen werden, können beim Aufruf der Konvertierungsmethode bereits alle benötigten Informationen zur Verfügung gestellt werden. Sollte eine Information (in diesem Fall ein Parameter) fehlen, kommt es bereits ganz am Anfang zu einer Ausnahme, auf die entsprechend reagiert werden kann. Bibliotheken und Implementierungen für den Zugriff auf Webservices sind in nahezu allen Programmiersprachen vorhanden, sodass die Implementierung eines entsprechenden Clients keinen großen Aufwand erfordert. Die verfügbaren Methoden werden über eine WSDL-Datei veröffentlicht.

4.2. Keine oder wenig Zwischenschritte

Die Anforderung „Keine oder wenig Zwischenschritte“ ergibt sich aus der Betrachtung des Ist-Zustands im letzten Kapitel und der daraus resultierenden Erkenntnis, dass viele Konverter nur unzureichend genau arbeiten.

4.2.1. Anforderung

Da jeder Konvertierungsvorgang verlustbehaftet ist, führt die Konvertierung über viele Zwischenschritte zwangsläufig zu einem schlechteren Umwandlungsergebnis. Es muss also versucht werden, die Anzahl der Zwischenschritte zu minimieren und bei der Pfadwahl darauf zu achten, die Pfade auszuwählen, bei denen der Verlust am geringsten ist, um den im letzten Kapitel beschriebenen „Stille Post“-Effekt zu vermeiden.

4.2.2. Lösungsansatz

Das Problem wird durch die Betrachtungen im folgenden Abschnitt noch erweitert und letztlich auch gelöst. Es bleibt jedoch zu sagen, dass bei einer Pfadwahl grundsätzlich nur die Pfade in Frage kommen, bei denen sowohl die Disk-Images, als auch die Hardwarespezifikation übernommen wird.

4.3. Konvertierung von jedem und in jedes Format

Die Forderung nach einem Universalkonverter ist gleichbedeutend mit der Anforderung „Konvertierung von jedem und in jedes Format“ und ist damit ähnlich wie der erste Punkt von zentraler Bedeutung für die Erfüllung der Aufgabenstellung.

4.3.1. Anforderung

Da der Konverter nicht bloß für bestimmte Anbieter und Produkte funktionieren soll, muss es die Möglichkeit geben, potentiell jedes beliebige Ausgangsformat in jedes beliebige Zielformat zu überführen. Die intuitive und naheliegendste Lösung für dieses Problem – nämlich einen Konverter für jede Quelle-Ziel-Kombination zu programmieren – erweist sich bei näherer Betrachtung als nicht praktikabel. Zum einen gibt es viel zu wenig Konvertierungspfade, die durch die Tools der Hersteller abgedeckt werden (siehe Abbildung 3.2), und zum anderen hängt die Anzahl der erforderlichen Konverter quadratisch von der Anzahl der unterstützten Produkte und Anbieter ab, da man jeweils einen Konverter für die Hin- und Rückrichtung benötigt (Abbildung 4.1).

Eine Implementierung nach diesem Schema ist also nicht möglich, da der Aufwand bei Änderungen oder Erweiterungen mit zunehmender Anzahl von unterstützten Produkten irgendwann nicht mehr vertretbar wäre.

4.3.2. Lösungsansatz

Zur Lösung dieser Problematik wählen wir ein geeignetes Zwischenformat, dass es als universelle Anlaufstation ermöglicht, von jeder beliebigen Quelle zu jedem beliebigen Ziel zu gelangen (Abbildung 4.2). Da die Konvertierung so immer nur über genau eine Zwischenstation läuft, sind die Genauigkeitsverluste bei der Umwandlung fast zu vernachlässigen. Auch der für die Implementierung erforderliche Aufwand wächst dadurch nur noch linear. Man benötigt nämlich je Anlaufstation nur noch zwei Konverter (hin und zurück). Dies führt dazu, dass jeder Konverter vollkommen unabhängig von den anderen ist, was eine zusätzliche Modularität gewährleistet.

Um eine solche Lösung zu implementieren, muss das gewählte Zwischenformat aber gewisse Grundvoraussetzungen erfüllen:

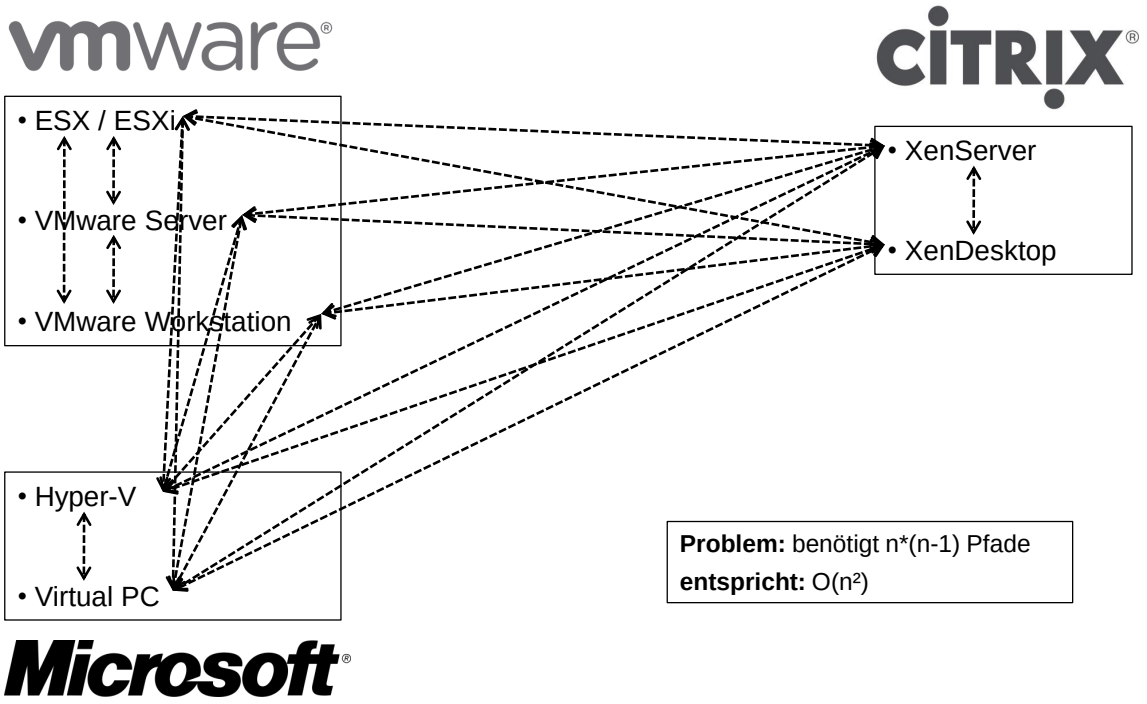

Abbildung 4.1.: Theoretisch benötigte Konvertierungspfade

- Das gewählte Format kann verlustfrei die Spezifikation aller Anbieter abbilden.
- Als Export- und Importformat wird es von möglichst vielen Herstellern unterstützt.
- Es existiert eine Implementierungsvorschrift, die eine saubere Umsetzung seitens der Hersteller und Konverterprogrammierer zulässt.

Bei der Suche nach einem geeigneten Format stößt man relativ schnell auf das Open Virtualization Format (OVF) [DMT10]. Es handelt sich dabei um das bereits in den Grundlagen beschriebene standardisierte Format der unabhängigen Distributed Management Task Force (DMTF), das derzeit den De-facto-Standard für die Bereitstellung sogenannter Virtual Appliances (d. h. vorkonfigurierter virtueller Systeme mit einem bestimmten Einsatzzweck) darstellt. Da die meisten Hersteller direkt oder indirekt (über einen Konverter) in der Lage sind, diese Virtual Appliances zu verarbeiten, bietet sich das Format als Zwischenformat geradezu an. Im OVF besteht eine Maschine aus einer XML-basierten Hardwarespezifikation und den Disk-Images, die in einem beliebigen Format vorliegen können. Da nicht alle Hersteller mit jeder Art von Disk-Image umgehen können, müssen wir später den Konvertierungsschritt in zwei Teile aufgliedern (siehe Kapitel 6).

4.4. Erhalt der Spezifikation

Viele Konverter verwerfen die Spezifikation der Maschine. Daher muss eine Lösung ersonnen werden, die es ermöglicht, die Spezifikation zu erhalten.

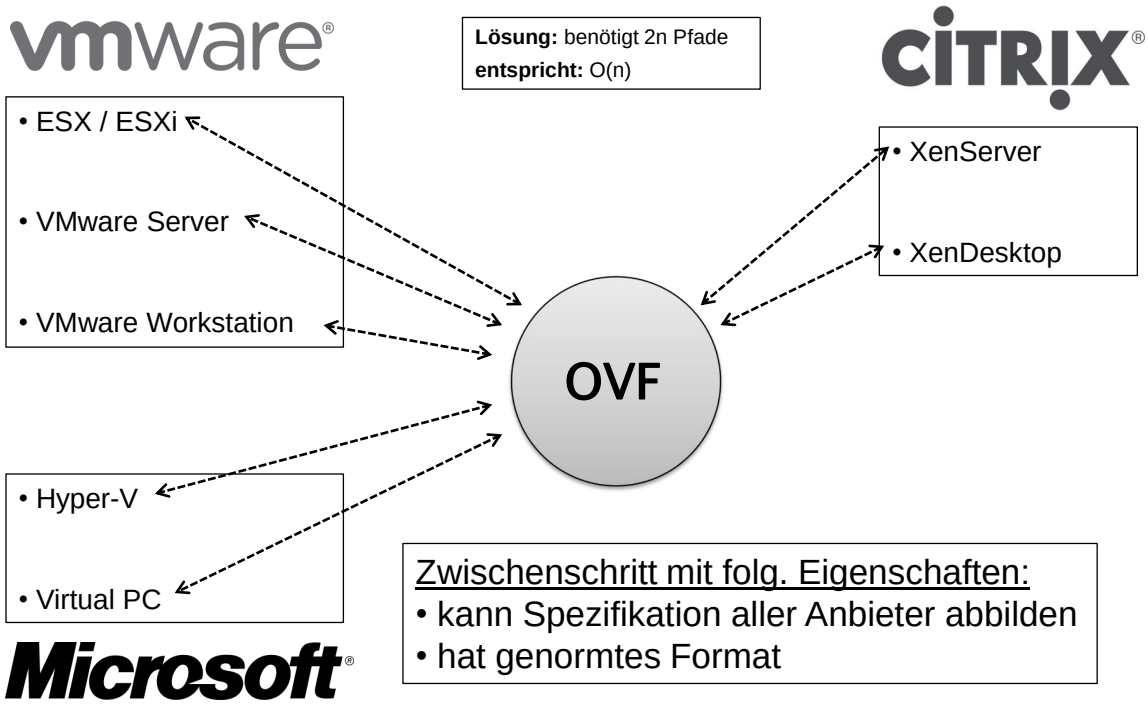

Abbildung 4.2.: Konvertierung über einen Zwischenschritt (OVF)

4.4.1. Anforderung

Da die Konvertierung über einen Zwischenschritt läuft und eine virtuelle Maschine ohne Hardwarespezifikation nicht lauffähig ist, muss sichergestellt werden, dass die Spezifikation der virtuellen Maschine möglichst verlustfrei erhalten bleibt. Ziel des ganzen Konvertierungsprozesses soll es letztlich sein, beim Umzug von virtuellen Maschinen die Einfachheit eines Umzugs einer physischen Maschine abzubilden, bei der man den Rechner (wie er ist) nur von einem Raum in den anderen zu tragen braucht. Bisherige Konvertierungsverfahren agieren eher so, als würde man den Rechner auseinanderbauen und die Einzelteile in einen neuen Raum tragen, um sie dort von einem Fremden wieder zusammenbauen zu lassen.

4.4.2. Lösungsansatz

Da man die Qualität und Arbeitsweise der Herstellerwerkzeuge nicht anpassen kann, muss man einen anderen Weg finden, die Validität der Spezifikation sicherzustellen. Wie im vorigen Abschnitt erwähnt, wird als Zwischenschritt das OVF benutzt, welches tendenziell mächtig genug ist, alle Informationen jedes beliebigen Herstellers abzubilden. Auch die Fähigkeit, OVF-Spezifikationen zu verstehen, ist bei den entsprechenden Werkzeugen bereits vorhanden. Die wichtigsten Punkte sind also durch die Wahl des Zwischenformats bereits abgedeckt und bedürfen keiner gesonderten Lösung. Trotzdem muss nach jedem Konvertierungsvorgang die Gültigkeit der Hardwarespezifikation überprüft und gegebenenfalls „repariert“ werden, da die Tools der Hersteller sich nicht 100%ig an die Spezifikation halten oder zu starken Gebrauch von den generischen Elementen machen (siehe Grundlagen).

Sollte es außerdem den Fall geben, bei dem ein Hersteller nicht in der Lage ist, mit OVF-

Dateien umzugehen, muss ein gesonderter Konverter für die Umwandlung der Spezifikation ins Open Virtualization Format (und zurück) manuell geschrieben werden.

4.5. Plattformunabhängigkeit und verteiltes Rechnen

Der Bedarf an verteiltem Rechnen und Plattformunabhängigkeit erwächst aus dem Fakt, dass unterschiedliche Konvertierungswerkzeuge zum Einsatz kommen, die unterschiedliche Anforderungen an ihr System haben.

4.5.1. Anforderung

Die Konvertierungswerkzeuge der Hersteller benötigen unterschiedliche Betriebssysteme und können somit nicht einfach auf einem einzigen System installiert werden, um hier als riesiges monolithisches Gebilde die Konvertierungsaufträge zu bearbeiten. Stattdessen ist es nötig, die Konverter auf unterschiedliche Systeme aufzuteilen. Der Modularität wegen bietet es sich an, für jeden Konverter (z. B. Xen $\leftrightarrow$ OVF) ein eigenes System aufzusetzen. Nun stellt sich natürlich die Frage, wie man die einzelnen unabhängigen Konverter verbindet, sodass sie auf sinnvolle Art und Weise zusammenarbeiten. Außerdem muss eine Lösung für den Austausch der auftragsbezogenen Daten gefunden werden, da die an einer Konvertierung beteiligten Konverter sich den Zugriff auf ein und dieselben Dateien (Quelldateien, Dateien im Zwischenformat, Dateien im Zielformat) teilen müssen.

4.5.2. Lösungsansatz

Da ohnehin eine Webservice-Schnittstelle für den Zugriff „von außen“ programmiert werden muss, können Webservices auch für die Kommunikation zwischen den Konvertern zum Einsatz kommen. Somit kann eine genormte Schnittstelle definiert werden, die für die Kommunikation verwendet wird, ohne dass zusätzlicher Programmieraufwand anfällt. Die zwischen den Arbeitsschritten auszutauschenden Daten werden über ein zentrales Netzlaufwerk bereitgestellt, das auf jedem Konvertersystem eingehängt wird. So brauchen zwischen den Teilschritten nicht erst umständlich Daten hin- und hergeschickt zu werden. Für die einzelnen Systeme bietet sich eine virtuelle Umgebung an, da hier eine ausreichend hohe Flexibilität vorhanden ist, um schnell und zuverlässig genügend eigenständige Systeme für die entsprechenden Konverter zur Verfügung zu stellen.

5. Systementwurf

Um die erarbeiteten Herausforderungen zu lösen, wird im folgenden Kapitel ein Rahmenwerk konzipiert, welches in der Lage ist die besprochenen Lösungsansätze umzusetzen.

5.1. Architektur

Das Rahmenwerk, welches die Aufgabe hat, die einzelnen Bestandteile des Systems zu verbinden, soll als Grundlage eine Model-View-Controller-Architektur nutzen. Jede der Konvertermaschinen betreibt dabei eine eigenständige Instanz des Rahmenwerks, die über das Agenten-Entwurfsmuster an einen zentralen Controller angebunden wird. Dieser Controller übernimmt die Aufgabe, die eingehenden Konvertierungsjobs zu planen und an die zuständigen Konverter weiterzuleiten. Ein Controller kann dabei auch mehrere Konverter für den selben Pfad betreuen, was es ermöglicht, ein Load-Balancing auf Jobebene durchzuführen. Zusätzlich wird durch diese Mehrfachbelegung von Konverterpfaden auch eine gewisse Redundanz erreicht, damit beim Ausfall eines Teilsystems das Gesamtsystem funktionsfähig bleibt. Die interne Kommunikation zwischen den Instanzen des Systems läuft dabei ähnlich wie die externe Schnittstelle über Webservices. Wie bereits in Abschnitt 4.5 beschrieben, wird jeder Maschine zusätzlich die Möglichkeit gegeben, über ein zentrales Netzlaufwerk auf gemeinsame Daten zuzugreifen, sodass ein umständlicher Austausch der Ergebnisse der entsprechenden Konvertierungsteilschritte entfällt (Abbildung 5.1).

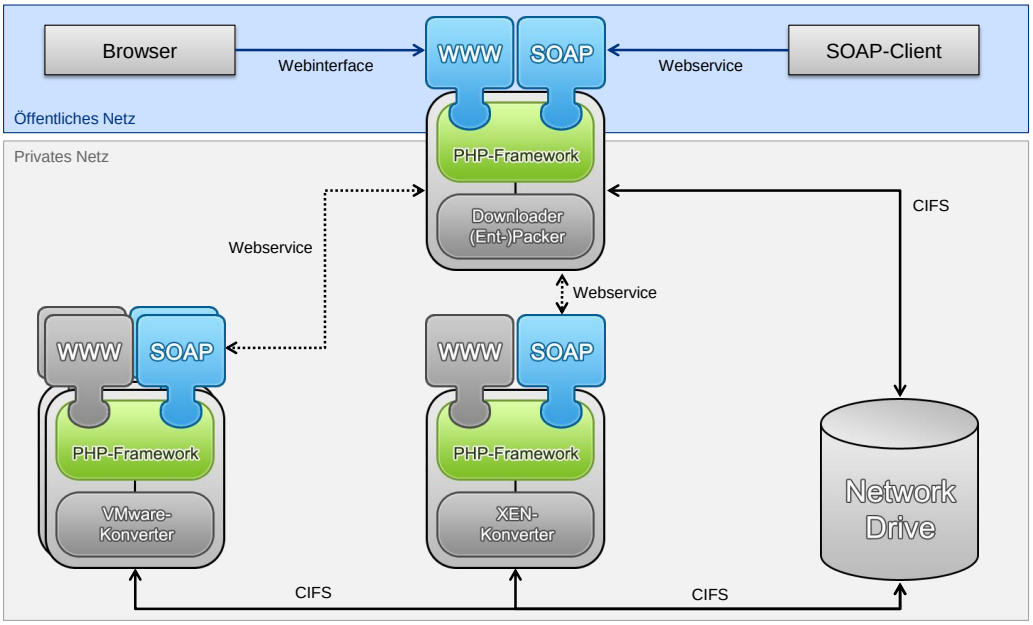

Abbildung 5.1.: Systemarchitektur

5.1.1. Das Rahmenwerk

Wie die Instanzen untereinander kommunizieren ist im obigen Abschnitt beschrieben. Wie sie ihre Schnittstellen veröffentlichen und intern eine Anfrage abarbeiten soll nun aber nochmal im Detail geklärt werden (vergleiche Abbildung 5.2).

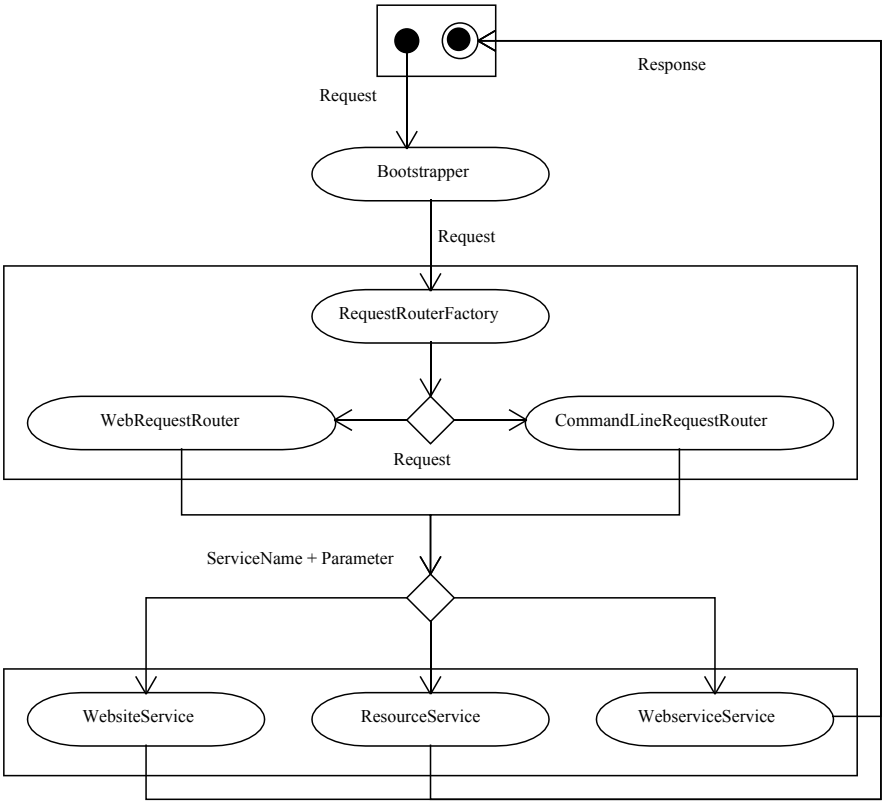

Abbildung 5.2.: Abarbeitung einer Anfrage im Rahmenwerk

Die Abarbeitung beginnt immer an der gleichen Stelle – bei einem zentralen Bootstrapper, der alle Anfragen entgegennimmt. Damit auch wirklich alle Anfragen beim Bootstrapper landen, wird der Webserver über Rewrite-Regeln umgelenkt. Diese zentrale Anlaufstelle hat die Aufgabe, das Rahmenwerk zu initialisieren und alle benötigten Dateien einzubinden. Zu den Hauptaufgaben zählt dabei die Einrichtung einer Fehlerbehandlungsroutine und das Registrieren eines Klassenloaders, der automatisch fehlende Klassen nachlädt. Sobald dies geschehen ist, nimmt das Rahmenwerk die Arbeit auf und prüft über eine sogenannte `RequestRouterFactory`, aus welcher Quelle die Anfrage stammt. Zur Zeit werden Kommandozeilenaufrufe (`CommandLineRequestRouter`) und Aufrufe aus dem Web (`WebRequestRouter`) unterstützt. Je nachdem aus welcher Quelle die Anfrage stammt, werden aus den Aufrufinformationen die übergebenen Parameter ermittelt. Der erste Parameter bestimmt dabei immer die gewünschte Service-Klasse. Ein Service gibt an, wie mit den erhaltenen Parametern verfahren wird. So lädt der `WebsiteService` z. B. eine Webseite, während der `WebserviceService` die Anfrage als SOAP-Anfrage interprätiert und versucht, die Parameter an eine freigegebene API-Klasse weiterzuleiten. Das Ergebnis der spezifischen Abarbeitung wird dann als Response an den Aufrufer zurückgegeben. Durch dieses Abarbeitungs-

schema können ganz unterschiedliche Schnittstellen über ein gemeinsames System angeboten werden. Jeder Service ist zudem komplett autark und kann so ganz spezielle Bearbeitungsroutinen und Entwurfsmuster implementieren. Der `WebsiteService` beispielsweise, der aus XML-Templates ein komponentenbasiertes Interface generiert, baut auf einem Model-View-Controller-Entwurfsmuster auf.

5.1.2. Genormte Schnittstellen

Damit unabhängige Entwickler eine Implementierungsvorschrift für die unterschiedlichen Klassen erhalten, werden vordefinierte Schnittstellen eingesetzt. Diese haben aber nicht nur die Aufgabe, als Schnittstellendefinition für die Programmierer zu dienen, sondern erfüllen zusätzlich den Zweck, generisch mit beliebigen Konvertern umgehen zu können.

5.2. Standardisierter Ablauf

Um die Lernkurve für Benutzer und Programmierer möglichst klein zu halten, soll sowohl das exemplarisch programmierte Web-Interface als auch die Programmierschnittstelle für unterschiedliche Konverterpfade immer auf die gleiche Weise funktionieren. Über das so standardisierte System wird aber nicht nur die Arbeit erleichtert, sondern unabhängigen Entwicklern zusätzlich auch die Möglichkeit eingeräumt, eigene Konverter zu implementieren. Zusammen mit den zuvor angesprochenen Schnittstellendefinitionen ergeben sich sehr klare Richtlinien, die definieren, wie ein Konvertierungsvorgang vollzogen werden muss, um mit dem System kompatibel zu sein. Um den Ablauf wie beschrieben zu standardisieren, betrachten wir im folgenden Abschnitt die zentralen Fragestellungen.

5.2.1. Wie werden die Quelldateien bereitgestellt?

Da die virtuellen Maschinen häufig aus mehreren Dateien bestehen (Disk-Images, Spezifikation usw.) bietet es sich an, die Daten zur einfacheren Verwaltung vorher in einem Archiv zusammenzufassen. Damit die für das Entpacken benötigte Rechenleistung nicht zu groß wird und die Verarbeitung schneller geht, soll hier auf ein Archivformat zurückgegriffen werden, das keine Komprimierung beinhaltet. Aufgrund der großen Verbreitung und Robustheit wird das TAR-Format als Archivierungsverfahren gewählt. Damit das Einstellen eines Jobs ohne Verzögerung vonstatten geht, werden dem Job die Archivinformationen nicht direkt, sondern in Form einer URL übergeben. So muss der Client nicht warten, bis die Quelldaten komplett übertragen sind, und kann direkt mit der Abarbeitung fortfahren. Stattdessen übernimmt der Konverter das Herunterladen der Quelldaten und vereinfacht so die Komplexität des Clients.

5.2.2. Wie werden die Konvertierungsergebnisse zur Verfügung gestellt?

Analog zu den Quelldateien bestehen auch die Konvertierungsergebnisse oft aus mehreren Einzeldateien, sodass auch bei der Bereitstellung der Einsatz eines Archivs von Nutzen ist. Logischerweise verwendet man hier das gleiche Format wie bei der Bereitstellung, da so die Ergebnisse einer Konvertierung gleich wieder als vollwertige Ausgangspunkte für weitere Konvertierungsvorgänge eingesetzt werden können. Da der Konverter nicht aktiv mit dem

Client kommuniziert, sondern als Dienstanbieter lediglich Aufträge entgegennimmt und abarbeitet, gibt es keine Möglichkeit, dem Client die Ergebnisse zu schicken. Stattdessen werden die Resultate unter einer automatisch generierten URL für eine gewisse Zeit veröffentlicht und können vom Client abgeholt werden. Diese Veröffentlichungsmethode erlaubt es analog zur Wahl des Archivformats, die Konvertierungsergebnisse wieder als Quelle zu verwenden.

5.2.3. Welche Informationen müssen beim Anlegen eines Jobs übergeben werden?

Um das Anlegen eines Jobs so intuitiv wie möglich zu gestalten und eine standardisierte Benutzung zu gewährleisten, wird darauf verzichtet, plattformspezifische Parameter abzufragen. Somit ergeben sich für das Anlegen eines Jobs nur die Parameter „URL zu den Quelldateien“, „Quellformat“ und „Zielformat“. Alle weiteren Informationen werden der Spezifikation entnommen.

5.2.4. Welche Schritte werden bei einer Konvertierung durchlaufen?

Wie in Abbildung 5.3 zu sehen ist, beginnt der Konvertierungsvorgang mit dem Herunterladen der Quelldateien. Anschließend wird das heruntergeladene Archiv entpackt und somit für die Arbeit des ersten Konverters vorbereitet. Sobald die Extraktion abgeschlossen ist, wird der für das Quellformat zuständige Konverter darüber informiert und beginnt mit der Konvertierung ins Open Virtualisation Format. Daraufhin wird der Zielkonverter benachrichtigt und nimmt seine Arbeit auf. Nachdem auch diese Umwandlung erledigt ist, werden die nun im Zielformat vorliegenden Daten gepackt und unter einer automatisch generierten URL veröffentlicht. Dem Client bleibt dabei die Aufgabe überlassen, in regelmäßigen Abständen zu prüfen, ob der eingestellte Job bereits abgearbeitet wurde.

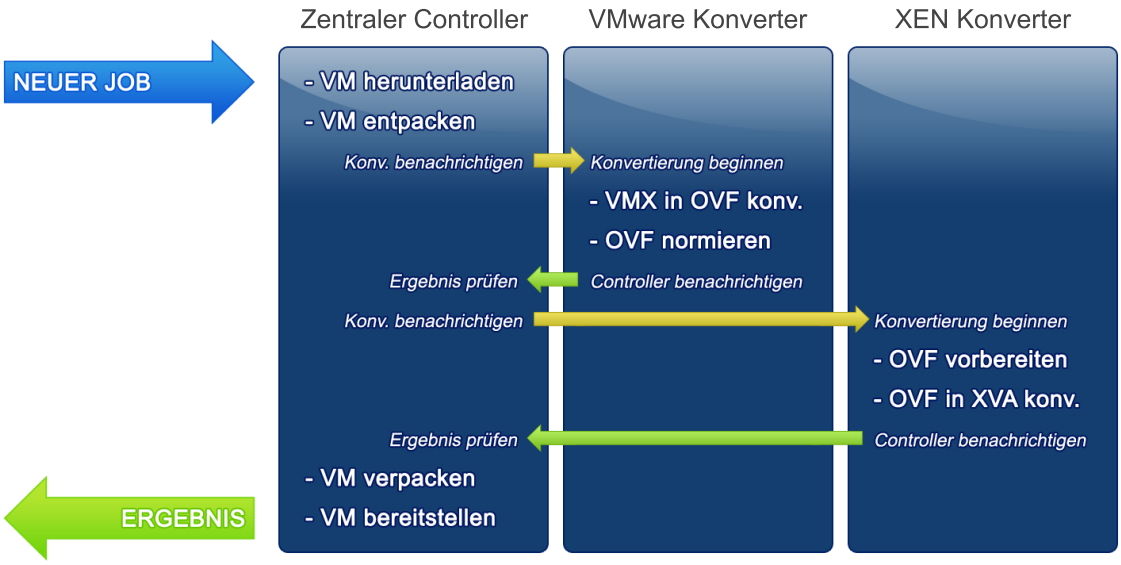

Abbildung 5.3.: Beispielablauf einer Konvertierung (VMware nach Xen)

6. Implementierung

Dieser Abschnitt der Arbeit geht den Entstehungsprozess der programmierten Anwendungen ein und beschäftigt sich mit den verwendeten Technologien.

6.1. Eingesetzte Softwarekomponenten

Am Anfang einer Implementierung steht immer die Frage, welche Software für die Lösung des Problems am besten geeignet ist. So muss man sich z. B. für eine Programmiersprache entscheiden oder festlegen, welche Datenbank zum Einsatz kommt. Auch die Wahl eines geeigneten Webservers muss in diesem Fall getroffen werden. Diese Entscheidungen müssen wohl überlegt sein, da eine Änderung im späteren Verlauf der Arbeit nur schwer möglich ist.

6.1.1. Programmiersprache

Da das Ziel der Arbeit die Erstellung eines interaktiven Webinterfaces und einer Webserviceschnittstelle ist, bieten sich webbasierte Skriptsprachen an, die eine schwache Typisierung besitzen. Die schwache Typisierung sorgt dafür, dass bei der Verwendung von Webservices nicht erst umständlich Client-Stubs für die Implementierung erstellt werden müssen, sondern generisch auf den Dienst zugegriffen werden kann.

Auch die Verbreitung der Programmiersprache spielt bei der Wahl eine zentrale Rolle. Es nützt nämlich nichts, wenn man eine Sprache einsetzt, die zwar gut für die Problematik geeignet, aber vollkommen unbekannt ist. Denn nur wenn es genügend Entwickler gibt, die in der Lage sind, mit der gewählten Sprache umzugehen, kann das Projekt wachsen. Gerade bei der Vielzahl an vorhandenen Virtualisierungslösungen ist man auf ein kollaboratives Arbeiten mit Third-Party-Entwicklern angewiesen.

Aufgrund eigener Erfahrungen, der großen Verbreitung und den oben genannten Punkten wird für dieses Projekt das kostenlose PHP [The] in der Version 5.3 eingesetzt.

6.1.2. Datenbank

Im Gegensatz zur Programmiersprache sind die Anforderungen an die Datenbank relativ gering, da nur wenige Informationen abgelegt werden müssen. Es wird lediglich verlangt, dass relationale Beziehungen abgebildet und Transaktionen verwendet werden können. Da PHP die größte Unterstützung für MySQL [Sun] bereit hält und MySQL unter Verwendung der Storage-Engine „InnoDB“ in der Lage ist, diesen Anforderungen zu genügen, wird als Datenbank das ebenfalls kostenlose MySQL eingesetzt.

6.1.3. Webserver

Entsprechend der oben getroffenen Entscheidungen muss der Webserver natürlich fähig sein, PHP-Skripte auszuführen. Zudem sollte er für den Einsatz auf unterschiedlichen Plattformen

(Linux und Windows) geeignet sein, da man, wie in den Anforderungen bereits umrissen, ein verteiltes System benötigt, um den unterschiedlichen Anforderungen der Konvertertools gerecht zu werden. Der bekannteste Webserver, der in Zusammenarbeit mit PHP oft zum Einsatz kommt, ist der Apache [Apa]. Da auch er kostenlos bezogen werden kann, und somit nur kostenlose Komponenten zum Einsatz kommen, die keine weitere Lizenzierungskosten benötigen, ergibt sich eine perfekte Grundlage für das kollaborative Arbeiten im Open-Source-Bereich.

6.1.4. Automatisierung

Zur Automatisierung der Herstellertools werden unterschiedliche Werkzeuge und Techniken eingesetzt, die im Folgenden kurz angesprochen werden.

Automatisierung unter Windows

Standardmäßig wird auf Batchdateien zur Automatisierung von Werkzeugen mit Kommandozeilenschnittstelle zurückgegriffen. Für die Automatisierung von Programmen mit grafischer Oberfläche wird AutoIt [Aut] eingesetzt. Dabei handelt es sich um ein Programm, das es erlaubt, Makros zu erstellen, die selbständig auf bestimme Ereignisse reagieren und somit eine robuste Möglichkeit bieten, fehlende Kommandozeilen-Funktionalitäten zu überbrücken. Die so erreichte Automatisierbarkeit ist jedoch sehr umständlich, da für jedes Programm ein eigenes Skript programmiert und gepflegt werden muss. Ein Beispiel-AutoIt-Skript zur Automatisierung des Starwind V2V-Konverters, welcher später zur Umwandlung von vhd- in vmdk-Disk-Images benutzt wird, findet man im Anhang in Listing A.1.

Automatisierung unter Linux

Da alle eingesetzten linuxbasierten Programme mit einer Kommandozeilenschnittstelle ausgestattet sind, werden unter Linux keine zusätzlichen Programme zur Automatisierung benötigt. Stattdessen werden, wie bei Linux üblich, Shell-Skripte eingesetzt um die Anwendungen zu bedienen.

6.1.5. Konverter

Exemplarisch werden die Konvertierungspfade „Citrix ↔ OVF“ und „VMware ↔ OVF“ implementiert. Da der VMware vCenter Converter für die Konvertierungsaufgabe nicht geeignet ist, wird auf die ovftools (ebenfalls von VMware) ausgewichen. Somit kommt für den erstgenannten Pfad der Citrix XenConvert und für den zweiten Pfad die ovftools zum Einsatz.

6.2. Unterschiede im Open Virtualization Format

Wenn man die im Zwischenformat generierten Dateien untersucht, stellt man fest, dass das OVF von vielen Herstellern nicht sauber implementiert wurde. So generiert Citrix XenConvert beispielsweise Hardwarespezifikationen, in denen Festplatten fälschlicherweise als ResourceType 19 codiert sind. Richtig wäre hier der ResourceType 17. Desweiteren bauen einige Hersteller Informationen wie z. B. BIOS-Einstellungen oder Bootdelays in die Spezifikation ein, die dort eigentlich nichts zu suchen haben. Das hat zur Folge, dass die generierten

OVF-Dateien nicht ohne weiteres weiterverarbeitet werden können. Damit herstellerfremde Anwendungen in der Lage sind, mit den Daten umzugehen, müssen sie vorher „repariert und normiert“ werden, sodass sie wieder der offiziellen Spezifikation genügen. Da die Defekte in den Dateien herstellerabhängig sind, kann man jedoch keine allgemeine Reparaturmethode implementieren, sondern benötigt eine auf den Konverter abgestimmte Prozedur.

Doch nicht nur beim Export gibt es Probleme. Auch beim Import bereits reparierter OVF-Dateien kann es zu Problemen kommen. So akzeptieren die ovftools z. B. nur Disk-Images im VMDK Format, sodass hier vor dem Import ebenfalls interveniert werden muss (Umwandlung über den Starwind V2V-Konverter – siehe oben).

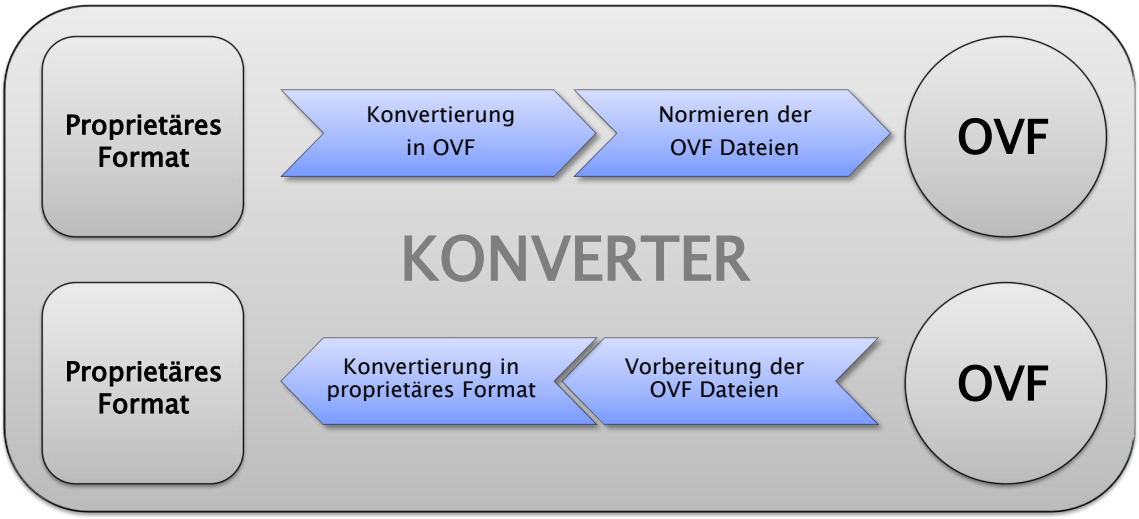

Abbildung 6.1.: Konvertierung in zwei Teilschritten

Um diese Problematik zu lösen, wird der Konvertierungsprozess sowohl auf dem Hin- als auch auf dem Rückweg in zwei Teilschritte unterteilt (Abbildung 6.1):

- Hinweg: proprietäres Format in OVF
  1. Ansteuern des Herstellerwerkzeugs zur Konvertierung der Quelldaten ins OVF-Format.
  2. Normieren der generierten OVF-Dateien und Beheben der herstellerspezifischen Fehler und Eigenheiten.
- Rückweg: OVF in proprietäres Format
  1. Vorverarbeiten der bereitgestellten OVF-Dateien und Herstellen der Kompatibilität.
  2. Ansteuern des Herstellerwerkzeugs zur Konvertierung der OVF-Daten ins Zielformat.

Wie die aus diesen Betrachtungen resultierende Schnittstellendefinition für einen Konverter aussieht, ist in Listing 6.1 zu sehen. Man erkennt hier deutlich, dass jeder Teilschritt der Konvertierung in einer separaten Methode abgelegt und somit optimal gekapselt ist.

```php
<?php
namespace seasons\vmTransfuse;

interface ConverterApi extends \seasons\core\api\Api {
  // Gibt den eindeutigen Namen des Konverters zurück
  public function getType();

  // Entfernt anbieterspezifische Reste aus dem Konvertierungsergebnis (Normalisierung)
  public function normalizeOvf($jobId);

  // Bereitet eine normalisierte OVF-Datei auf den Konverter vor
  public function prepareOvf($jobId);

  // Automatisiert die Konvertierung ins OVF-Format
  public function convertToOvf($jobId);

  // Gibt den Status der Konvertierung ins Zwischenformat zurück
  public function getOvfConversionState($jobId);

  // Gibt den Status der Konvertierung ins proprietäre Format zurück
  public function getNativeConversionState($jobId);

  //Automatisiert die Konvertierung ins proprietäre Format
  public function convertToNative($jobId);
}
?>
```

Listing 6.1: Interface für die Konverter

6.3. Ergebnisse

Nachdem die Grundlagen und Hintergründe in der bisherigen Ausarbeitung bereits hinreichend thematisiert wurden, soll nun im letzten Abschnitt dieser Arbeit auch kurz das Ergebnis vorgestellt werden.

6.3.1. Webserviceschnittstelle

Das Codebeispiel in Listing 6.2 zeigt, wie die fertige Webserviceschnittstelle verwendet wird und wie ein Client Konvertierungsaufträge beim Konverter einreichen und verwalten kann.

```php
<?php
include 'Api.php';

// Verbindung zum Konverter aufbauen
$api = Api::connect('projnm01.lab.ifi.lmu.de', 'username', 'password');

// Neue Konvertierungen starten
$jobId1 = $api->addJob('http://cip.ifi.lmu.de/~moogh/xen.tar', 'Xen', 'VMware');
$jobId2 = $api->addJob('http://cip.ifi.lmu.de/~moogh/ovf.tar', 'OVF', 'VMware');

```

```
11 // Laufende Jobs auslesen
12 foreach($api->getJobs() as $job) {
13   // Ergebnisse abgeschlossener Konvertierungen herunterladen
14   if($job['state'] == 'DONE') exec('wget ' . $job['publishmentUrl']);
15 }
16
17 // Existierenden Job abbrechen
18 $api->cancelJob($jobId1);
19 ?>
```

Listing 6.2: Webservice-Schnittstelle

Zuerst wird dabei die Hilfsklasse eingebunden, die den SoapClient repräsentiert. Anschließend wird eine Verbindung zum Konverter aufgebaut, bei der neben dem Hostnamen des Konverters auch die Zugangsdaten übergeben werden. Sollte das Framework nicht im Dokument-Root installiert sein, kann alternativ auch eine direkte Adresse zur WSDL-Datei statt dem Hostnamen übergeben werden. Sobald die Verbindung aufgebaut wurde, kann mit der Verwaltung der Jobs begonnen werden. So ist es beispielsweise möglich, neue Jobs anzulegen, alte zu löschen oder ganz einfach den Status bestimmter Vorgänge zu prüfen. Im Beispiel wird außerdem das Ergebnis eines bereits abgeschlossenen Konvertierungsvorgangs heruntergeladen.

6.3.2. Webinterface

Ähnlich wie bei der Benutzung der SOAP-Schnittstelle benötigt man für die Arbeit mit dem Konverter auch im Webinterface zunächst eine Authentifizierung. Sobald man seine Benutzerinformationen eingegeben und sich eingeloggt hat, gelangt man zur Startseite. Hier kann der Administrator des Systems den Nutzer über neue Funktionen oder Änderungen informieren. Im linken Bereich des Bildschirms befindet sich das Hauptmenü, welches die Möglichkeit bietet, zum gewünschten Programmpunkt zu springen. Neben der Option sich auszuloggen oder seinen Account zu verwalten, kann man hier auch zum Konvertermodul wechseln. Dort wird zunächst eine Liste aller bereits existierender Konvertierungsaufträge angezeigt (Abbildung 6.2). Man kann in dieser Liste den Status der einzelnen Aufträge nachvollziehen und bereits abgeschlossene Aufträge durch einen Klick auf die betreffende Zeile herunterladen.

Durch das Anwählen des Menüpunkts „Umwandlung starten“ gelangt man auf eine Seite, auf der sich neue Jobs anlegen lassen (Abbildung 6.3).

Nach dem Anlegen des Auftrags bekommt der Nutzer eine kurze Rückmeldung und das System reiht den Auftrag in die Warteschlange ein. Der Status kann nun wie gehabt über die Jobliste abgefragt werden (Abbildung 6.4). Nach abgeschlossener Arbeit kann der Nutzer das Ergebnis herunterladen und auf den gewünschten Zielserver verteilen.

Wie man sieht, ist die Benutzung relativ einfach und bedarf kaum Vorwissen oder Informationen über die herstellerspezifischen Aspekte der Virtualisierung. Das in der Aufgabenstellung definierte Ziel konnte also ohne allzu große Kompromisse realisiert werden.

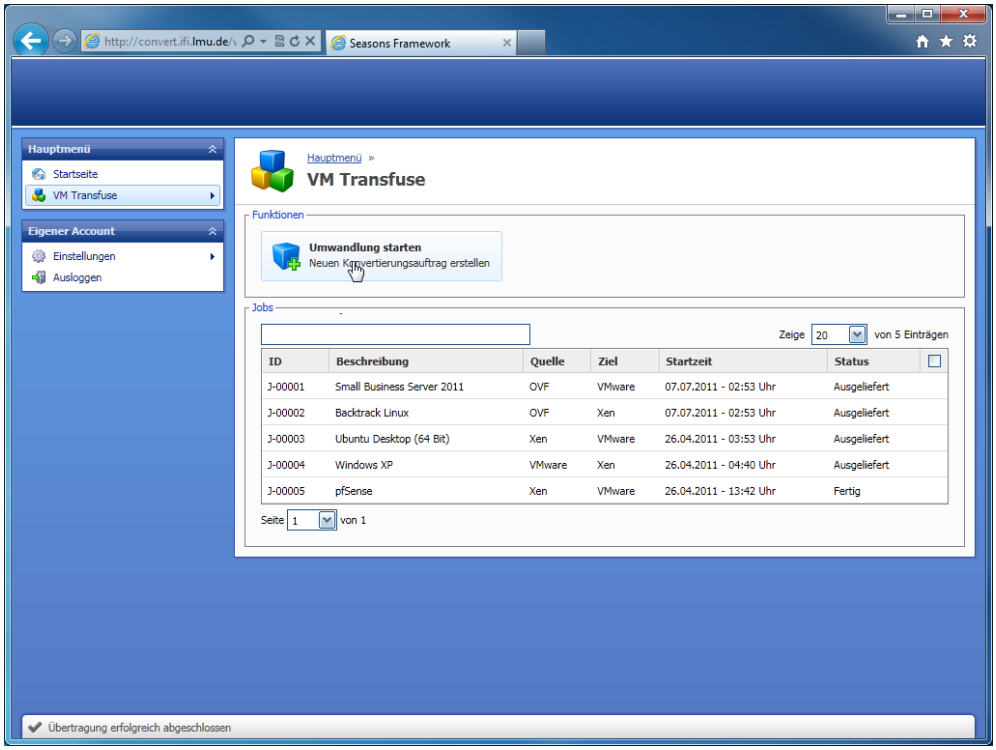

Abbildung 6.2.: Liste mit den Konvertierungsaufträgen

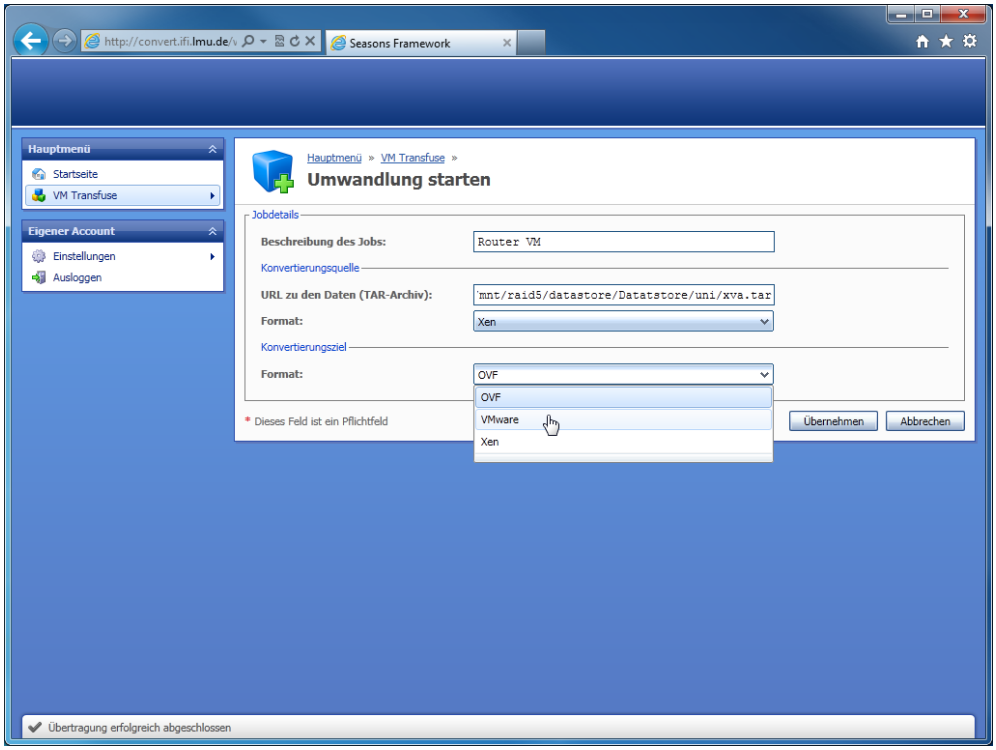

Abbildung 6.3.: Interface zum Anlegen eines Konvertierungsauftrags

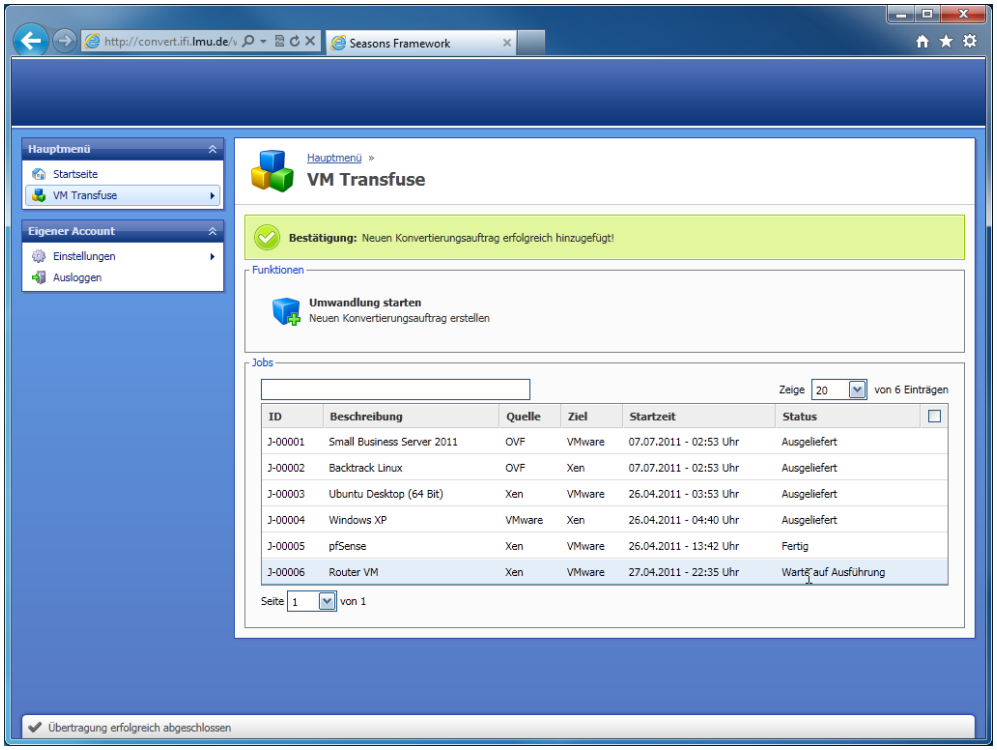

Abbildung 6.4.: Job erfolgreich angelegt

7. Fazit und Ausblick

Die Ergebnisse dieser Arbeit zeigen eindeutig, dass die automatische Konvertierung von virtuellen Maschinen möglich und sinnvoll ist. Es handelt sich bei den fehlenden Konvertierungsmöglichkeiten der Herstellerwerkzeuge also offensichtlich nicht um eine technische, sondern vielmehr um eine strategische Hürde, die künstlich seitens der Hersteller aufrecht erhalten wird. Die Intention hinter dieser Herangehensweise hat wahrscheinlich marketingtechnische Aspekte und zielt darauf ab, möglichst viele Kunden an die eigenen Produkte zu binden.

Diese Entwicklung hat negative Auswirkungen auf die weitere Verbreitung der Technologie und sollte überdacht werden. Man kann nur hoffen, dass die Hersteller in Zukunft weniger über ihre Marktdominanz, sondern vielmehr über neue und bessere Funktionalitäten versuchen werden, die Kunden für das eigene Produkt zu gewinnen.

7.1. Ausblick

Um die Funktionalität des programmierten Systems zu erweitern, können weitere Konverterpfade implementiert werden. Desweiteren wäre es denkbar, dass man eine direkte Schnittstelle zu den unterschiedlichen Plattformen integriert, die es erlaubt, die Maschinen direkt von einem Quell-Server zu beziehen oder auf einem Ziel-Server bereitzustellen, wo die Maschinen nach erfolgreichem Umzug auch automatisch eingeschaltet werden könnten. Somit würde man sich das manuelle Bereitstellen und Verteilen sparen. Auch die Erstellung einer plattform- und anbieterübergreifenden Management-Suite für den Umzug von virtuellen Maschinen wäre dadurch relativ einfach möglich.

A. Anhang

```
1  If $CmdLine[0] = 0 Then
2    MsgBox(0, "Command Line Usage:", "vhd2vmdk.exe <path to vhd file>")
3    Exit
4  EndIf
5
6  Run("StarV2V.exe")
7
8  WinWait("StarWind Software V2V Image Converter")
9  WinActivate("StarWind Software V2V Image Converter")
10 ControlClick ("StarWind Software V2V Image Converter", "Welcome to the
       StarWind Software image", "Button2")
11 WinWait("StarWind Software V2V Image Converter", "Select source file:")
12 WinActivate("StarWind Software V2V Image Converter", "Select source file:")
13 ControlClick ("StarWind Software V2V Image Converter", "Select source file
       :", "Button1")
14 WinWait("Öffnen", "")
15 WinActivate("Öffnen", "")
16 ControlSend("Öffnen", "", "Edit1", $CmdLine[1])
17 ControlClick ("Öffnen", "", "Button2")
18 WinWaitClose("Öffnen", "")
19 WinWaitActive("StarWind Software V2V Image Converter", "Select source file
       :")
20 ControlClick ("StarWind Software V2V Image Converter", "Select source file
       :", "Button3")
21 WinWait("StarWind Software V2V Image Converter", "VMWare Workstation vitual
        disk image (VMDK). Disk")
22 WinActivate("StarWind Software V2V Image Converter", "VMWare Workstation
       vitual disk image (VMDK). Disk")
23 ControlClick ("StarWind Software V2V Image Converter", "VMWare ESX server
       image", "Button11")
24 WinWait("StarWind Software V2V Image Converter", "Virtual disk type")
25 WinActivate("StarWind Software V2V Image Converter", "Virtual disk type")
26 ControlCommand ("StarWind Software V2V Image Converter", "SCSI", "Button3",
        "Check")
27 ControlClick ("StarWind Software V2V Image Converter", "Virtual disk type",
        "Button14")
28 WinWait("StarWind Software V2V Image Converter", "Select output file:")
29 WinActivate("StarWind Software V2V Image Converter", "Select output file:")
30 ControlClick ("StarWind Software V2V Image Converter", "Select output file
       :", "Button15")
31 WinWait("Done - StarWind Software V2V Image Converter", "Burning Log")
32 WinActivate("Done - StarWind Software V2V Image Converter", "Burning Log")
33 ControlClick ("Done - StarWind Software V2V Image Converter", "Finish", "
       Button16")
```

Listing A.1: AutoIt-Skript zur Automatisierung des Starwind V2V-Konverters

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[Apa] Apache Software Foundation: *Apache Webserver*. http://httpd.apache.org/.

[Aut] AutoIt Consulting Ltd.: *AutoIt*. http://www.autoitscript.com/site/.

[Cit10] Citrix: *Citrix XenConvert Guide*, April 2010. http://support.citrix.com/servlet/KbServlet/download/20644-102-332133/XenConvertGuide.pdf.

[DMT10] DMTF: *Open Virtualization Format Specification*, Januar 2010. http://www.dmtf.org/sites/default/files/standards/documents/DSP0243_1.1.0.pdf.

[ITW11] ITWissen.info: *Virtualisierung :: virtualization technology :: VT :: ITWissen.info*, April 2011. http://www.itwissen.info/definition/lexikon/Virtualisierung-VT-virtualization-technology.html, aufgerufen am 21.04.2011.

[Mic10a] Microsoft: *Neues zu virtuellen Festplatten*, Oktober 2010. http://technet.microsoft.com/de-de/library/dd440864%28WS.10%29.aspx, aufgerufen am 17.04.2011.

[Mic10b] Microsoft: *V2V: Konvertieren virtueller Maschinen in VMM*, Juli 2010. http://technet.microsoft.com/de-de/library/cc793147.aspx, aufgerufen am 23.04.2011.

[Ora11] Oracle Corporation: *Oracle VM VirtualBox – User Manual: Chapter 5. Virtual storage*, April 2011. http://www.virtualbox.org/manual/ch05.html.

[QEM08] QEMU Developer: Mark McLoughlin: *The QCOW2 Image Format*, September 2008. http://people.gnome.org/~markmc/qcow-image-format.html.

[Ste10] Stephens, Bob: *Diagram & Icon Library - Community 1 of 2*, September 2010. http://communities.vmware.com/servlet/JiveServlet/download/13702-1-43760/VMW_10Q3_PPT_Library_VMware_icons-diagrams_R7_COMM_1_of_2.pptx, aufgerufen am 17.04.2011.

[Sun] Sun Microsystems GmbH: *MySQL*. http://www.mysql.de/.

[The] The PHP Group: *PHP*. http://php.net/.

[VMw07] VMware: *VMware Virtual Disks – Virtual Disk Format 1.1*, November 2007. http://www.vmware.com/app/vmdk/?src=vmdk.

[VMw11] VMware: *VMware vCenter Converter Installation and Administration Guide*, März 2011. http://www.vmware.com/pdf/vsp_vcc_42_admin_guide.pdf.

[Wor07] World Wide Web Consortium: *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)*, April 2007. `http://www.w3.org/TR/soap12-part1/`.