Technische Fakultät
Albert-Ludwigs-Universität, Freiburg
Lehrstuhl für Kommunikationsysteme
Prof. Dr. Gerhard Schneider

Bachelorarbeit

Assisted Create View for Digital Artefacts

13. März 2012

Javier Castillo
Matr.-Nr.: 2554987

betreut durch
Dr. Dirk von Suchodoletz
Klaus Rechert

Erstprüfer
Prof. Dr. Gerhard Schneider

Zweitprüfer
Prof. Dr. Christian Schindelhauer

Erklärung

Hiermit erkläre ich, dass ich diese Abschlussarbeit selbständig verfasst habe, keine anderen als die angegebenen Quellen/Hilfsmittel verwendet habe und alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen wurden, als solche kenntlich gemacht habe. Darüber hinaus erkläre ich, dass diese Abschlussarbeit nicht, auch nicht auszugsweise, bereits für eine andere Prüfung angefertigt wurde.

Ort, Datum Unterschrift

Danksagung

Ich bedanke mich ganz herzlich bei meiner Frau Sarah. Sie stand mir immer unterstützend zur Seite und kümmerte sich liebevoll darum, dass ich mich in Deutschland wohlfühle und meine Ziele erreiche.

> *Las decisiones eran sólo el comienzo de algo. Cuando alguien tomaba una decisión, en realidad estaba sumergiéndose en una corriente poderosa, que lleva a la persona a un lugar que nunca había soñado a la hora de decidir.*
>
> *(Der Alchimist von Paulo Coelho)*

Mein Dank geht auch an Katja Messemer und Dirk von Suchodoletz, die mir immer wieder zeigten, dass Professionalität sehr eng mit Menschlichkeit und Freundlichkeit verbunden ist.

Zuletzt, aber nicht weniger herzlich, bedanke ich mich bei Ofelia Rassner und meinen ehemaligen Arbeitskollegen im Kulturverein Roccafé, die eine besondere Rolle bei meiner Integration in die deutsche Gesellschaft gespielt haben.

Inhaltsverzeichnis

1. Einleitung

Die Frage wie Information verewigt werden kann, beschäftigt die Menschheit seit mehreren Jahrhunderten. Heutzutage betrifft diese Frage besonders die Information in digitaler Form. Allein zwanzig Jahre alte Daten lassen sich nicht auf den aktuellen Betriebssystemen öffnen bzw. ausführen, ohne dass Darstellungsfehler auftreten. In vielen Fällen gehört ein Dateiformat zu einer bestimmten Anwendungen. Wie lang dieses Format existieren wird, hängt nicht nur von seinen eigenen Eigenschaften ab sondern auch von der Effizienz und Popularität dessen dazugehörigen Anwendung. Auch Hardware spielt hier eine große Rolle. Manche Anwendungen sind extra für eine bestimmte Rechnerarchitektur erschaffen und sobald diese Architektur nicht mehr Bestandteil der neuen Hardware ist, werden diese Anwendungen und viele ihrer Datenformaten obsolet.

Institutionen wie *OASIS*[1] schlagen präventive Lösungen wie *OASIS Open Document Format for Office Applications*[2] vor, eine Standardisierung für XML-basierten Dateiformate für Office-Anwendungen. Mit der Einführung eines offenen Formates soll erreicht werden, dass Office-Dokumente anwendungsunabhängig werden. Solang die Weiterentwicklung von Office-Anwendungen keine Änderung des Formats ihrer Dateien impliziert, kann man theoretisch die dauerhafte Verfügbarkeit solcher Dateien gewährleisten. Ein anderes Beispiel ist das Dateiformat *PDF/A*, das von der *ISO*[3] unter der Kennung *„ISO 19005-1:2005“* extra für die Langzeitarchivierung standardisiert wurde.[4]

> *„Ziel des PDF/A-Standards ist, dass PDF-Dokumente erstellt werden können, deren visuelles Erscheinungsbild über die Zeit erhalten bleibt. Dabei sollen diese Dateien unabhängig sein von Software und Systemen zu Herstellung, Speicherung und Reproduktion.“* [Drümmer u. a. 2007, S.9]

Beide o.g. Beispiele zeigen wie mit Hilfe offener Formate der Aufwand der Langzeitarchivierung in der Zukunft wesentlich reduziert werden kann. Obwohl diese Lösung

[1] Organization for the Advancement of Structured Information Standards
[2] International Organization for Standardization - International Electrotechnical Commission unter der Bezeichnung ISO/IEC 26300:2006
[3] International Organization for Standardization
[4] Vgl. [Drümmer u. a. 2007, S.9]

in der Aktualität nicht für jedes digitale Objekt, kurz DO, praktikabel ist, trägt diese mit sich ein Konzept, das für zukünftige Entwicklungen ausschlaggebend sein könnte.

Es gibt außerdem ein Referenzmodell für die Langzeitarchivierung digitaler Objekte namens *OAIS*[5]. Das unter *„ISO 14721“* definierte Referenzmodell hat sich weltweit durchgesetzt. In Abbildung 1.1 wird die Struktur des Modells als Diagramm dargestellt. Dabei wird der Weg des vom Produzenten erstellten digitalen Objektes in das Archivierungssystem aufgezeigt.

Abbildung 1.1.: Das OAIS-Referenzmodell [Neuroth u. a. 2009, Kap. 4.10]

OAIS verzichtet auf die Beschränkung von bestimmten Datentypen, Datenformaten oder Systemarchitekturen und strebt dabei anwendungsfähig und skalierbar sein für viele Institutionen und ihre jeweiligen Bedürfnisse[6].

Falls bestimmte digitale Objekte bereits obsolet sind, kommen wiederherstellende Lösungen wie *GRATE* zum Einsatz. *GRATE* steht für Global Remote Access To Emulation-Services und wurde am Lehrstuhl für Kommunikationssysteme der Universität Freiburg in Rahmen des PLANETS-Projekts[7] entwickelt. Der Java-basierte prototypische Webdienst versucht den Typ eines hochgeladenen digitalen Objektes zu identifizieren, um nach Ressourcen in seiner Datenbank zu suchen, die das Arbeiten mit dem DO ermöglichen. Die Ressourcen sind typischerweise Betriebssysteme, Anwendungen und Emulatoren, die zusammen eine emulierte Umgebung erstellen, in der das Arbeiten mit dem DO möglich ist.

[5] Open Archival Information System
[6] Vgl. [Neuroth u. a. 2009, Kap. 4:5]
[7] Preservation and Long-term Access through Networked Services. Siehe http://www.planets-project.eu/about/

GRATE zeigt, dass Emulation eine sehr gute Alternative ist, um die Umwandlung des veralteten DO-Formats in ein aktuelles Format zu umgehen. Die Darstellung bzw. Ausführung des digitalen Objektes in seiner ursprünglichen Erstellungsumgebung erhöht die Authentizität des Objektes und verbessert seine Verwendbarkeit.

Auf Grund der großen Anzahl an Betriebssysteme und ihrer Veralterung kann man nicht davon ausgehen, dass jeder durchschnittliche Computeranwender in der emulierten Umgebung zurechtkommt. Deswegen muss nach Lösungen gesucht werden, die es erlauben, automatisch auf das DO zuzugreifen. Damit die Lösungen allgemeingültig sind, ist der Objektzugriff über eine gewisse Abstraktionsschicht vorausgesetzt. Nur so kann das Automatisierungsverfahren für jede emulierte Umgebung benutzt werden, ohne von dieser abhängig zu sein.

Diese Arbeit beschäftigt sich mit dem abstrakten Ansatz namens *Assisted Create View*, der durch die Aufzeichnung und das Abspielen von Maus- und Tastatureingaben über das Client/Server-Protokoll VNC die Automatisierung verschiedener Routinen ermöglicht. Bestandteil dieses Ansatzes ist der von vielen Emulatoren bereitgestellte VNC-Server, der die Steuerung der emulierten Umgebung über VNC erlaubt ohne, dass der VNC-Server innerhalb derer läuft. Diese Besonderheit macht den Ansatz auf viele Systeme anwendbar.

1.1. Motivation

Umgebungen für das Arbeiten mit digitalen Objekten sind in Form von Containerdateien gespeichert. Diese werden System-Images genannt und benötigen einen Emulator, um die Umgebung zu erstellen. Neben den Automatisierungsansätzen, die Änderungen der System-Images oder die Installation zusätzlicher Tools in den Umgebungen benötigen, bietet der Ansatz *Assisted Create View* eine Möglichkeit, Arbeitsabläufe zu automatisieren ohne jegliche Angriffe innerhalb der emulierten Umgebung.

Der Ansatz benötig die Verwendung einer im Emulator integrierten VNC-Schnittstelle, die von den emulierten Umgebung völlig Unabhängig ist. Die Aufzeichnung und das Abspiele von Maus- und Tastatureingaben über VNC sollen die Basis für die Automatisierung von Routinen sein. Gespeicherte Routinen für ein bestimmtes DO-Typ sollen für viele weitere digitale Objekte gleicher Typ verwendbar sein. Die Reproduktion von Routinen für das Öffnen eines digitalen Objektes mit einer bestimmten Anwendung in der emulierten Umgebung oder die Anpassung von Einstellungen vor der eigentlichen Arbeit mit einer Anwendung sind Aufgabe, die durch eine Implementierung des Ansatzes getestet und analysiert werden sollen.

1. Einleitung

2. Grundlagen

2.1. Hardware-Emulation

Als Hardware-Emulation wird die durch Software Nachbildung von Hardware bezeichnet. Ein solches Software wird Emulator genannt und ermöglicht die Erstellung von virtuellen Maschinen. Eine virtuelle Maschine, kurz VM, ist die virtuelle Abbildung einer bestimmten Hardware, auf der die Installation von Firmwaren und/oder Betriebssystemen möglich ist. Der Rechner auf dem der Emulator angesiedelt ist, wird als Host-Rechner bezeichnet und das in der VM installierte Betriebssystem als Gastsystem. Die Aufteilung der echten (physikalische) Ressourcen, die für eine VM zur Verfügung stehen, wird durch einen sogenannten Hypervisor erledigt. Die meisten neuen Prozessoren unterstützen Virtualisierung auf Hardwareebene und erlauben somit der VM den direkten Zugriff auf die echte Hardware. Dadurch wird den Hypervisor entlastet und die Leistung der VM verbessert.

Einige bekannte Emulatoren sind bspw. QEMU[1], und Dioscuri[2] aus der Open-Source Branche und VMWare[3] und Virtualbox[4] aus der kommerzielle Branche. Die Verwendung von Open-Source Varianten spielt eine besondere Rolle bei der Implementierung neuer Ansätze. Ein nicht kommerzielles Lizenzierungsmodell und die Verfügbarkeit des Quellcodes des Emulators bietet eine große Freiheit bei neuen Entwicklungen und trennt sie von kommerziell gebundenen Lösungen, die die Entstehung von offenen Standards einschränken könnten.

Unter den o.g. Emulatoren zeichnet sich QEMU für seine zahlreichen Features und Popularität aus. Er läuft mit der Technik namens "Kernel-based" annähernd mit nativer Geschwindigkeit und emuliert auch Prozessorarchitekturen wie PowerPC oder ARM. Dioscuri hat die Besonderheit, dass er ursprünglich für den Einsatz in der funktionalen Langzeitarchivierung entwickelt wurde, sein modularer Aufbau und

[1] Siehe http://qemu.org
[2] Ein Emulator für die Langzeitarchivierung. Siehe http://dioscuri.sourceforge.net/
[3] Siehe http://www.vmware.com
[4] Siehe https://www.virtualbox.org/

seine auf Java basierte Entwicklung zeigen die Notwendigkeit der Verwendung von plattformunabhängige und anpassungsfähige Werkzeuge.

Einige Vorteile der Emulation in der Langzeitarchivierung sind[5]

- Der Aufwand je Dokument ist gering.
- Es wird nur ein Emulator für viele Dokumente gebraucht.
- Hohe Authentizität des Dokuments.
- Der laufende Aufwand ist proportional zur Verwendungshäufigkeit.

2.2. VNC

Virtual Network Computing, kurz VNC, ist eine Plattformunabhängige Client-Server-Anwendung für die Fernsteuerung von Rechnern. Die Arbeitsweise von VNC ist sehr intuitiv. Auf dem zu steuernden Rechner läuft ein VNC-Server, der durch das RFB-Protokoll[6] einem VNC-Client die Übertragung von Bildschirminhalten und Benutzereingaben ermöglicht.

Kein Austausch von Dateien zwischen Server und Client und die Übertragung von besonderen Tastenkombinationen vom Client zum Server waren früher einige Einschränkungen von VNC. Allerdings wurden solche Einschränkungen von VNC-Distributionen wie TightVNC[7] überwunden.

2.3. Emulatoren und VNC

Ein sehr komfortables Feature von manchen Emulatoren ist die Bereitstellung eines VNC-Servers für die Steuerung der VMs. Der VNC-Server befindet sich nicht in der VM sondern ist Teil des Emulators. Dadurch wir die VM nicht beeinträchtigt, sonst könnte ihre Performance beeinflusst werden.

[5] Vgl. [Höfler 2008, Seite 28]

[6] Remote Framebuffer Protokoll. Ein Netzwerkprotokoll für den Zugriff auf die grafischen Benutzungsoberflächen anderer Rechner. Siehe http://www.realvnc.com/docs/rfbproto.pdf

[7] Kostenlose Software zur Steuerung entfernter Rechner als Weiterentwicklung von VNC mit Komprimierung von Grafiken. Siehe http://www.tightvnc.com/

2.4. Datentransport in emulierten Umgebungen

Die Erstellung virtueller Umgebungen bezweckt die Benutzung von digitalen Objekten, die in vorhandenen aktuellen Systemen nicht möglich ist. Für den Transport von digitalen Objekten in eine VM kommen meistens folgende Strategien zum Einsatz:

Shared Folders

Shared Folders (dt. Freigegebene Ordner) ist nichts anderes als die Freigabe von Ordnern in einem Netzwerk. Die freigegebenen Ordner können sich sowohl auf einem entfernten Rechner als auch auf dem Rechner befinden, auf dem die VM läuft. Durch Protokolle wie SMB[8] erhält die VM zugriff auf den Ordner, in dem I/O-Operationen (solang vom Nutzer erlaubt) möglich sind. Die Verwendung von *Shared Folders* setzt die Unterstützung der notwendigen Protokolle so wie ein bereits eingerichtete Netzwerk-Umgebung in der VM voraus. Manche Emulatoren ermöglichen eine automatische Einbindung von vordefinierten freigegebenen Ordnern in einer emulierten Umgebung. Allerdings ist dieses Feature kein Standard und somit auch keine allgemeine Lösung für den Transport von Daten in die Emulation.

Virtuelle Datenträger

Auch Datenträger können innerhalb einer VM emuliert werden. Solche emulierte Datenträger können den Inhalt von verschiedenen Speicherorten des Hostsystems enthalten und werden in der VM so eingehängt, als ob diese echte Datenträger wären. Wichtig bei der Erstellung von virtuellen Datenträgern ist, dass diese mit einem von der VM unterstützten Dateisystemen[9] formatiert sind. Im Gegensatz zu *Shared Folders* benötigt die VM mit diesem Ansatz kein Netzwerk oder zusätzliche Software. Falls ein Betriebssystem ein bestimmtes Dateisystem nicht unterstützt, können Container mit unterschiedlichen Dateisystemen verwendet werden.

Ein Container ist die Abbildung des Dateninhaltes von einem Datenträger wie z.B. von einer Festplatten oder einer Floppy-Disk. Die Containers sind mit einem Dateisystem formatiert, welches von der VM erkannt wird. Wird ein Dateisystem von einer VM nicht unterstützt, so kann man einfach einen Container mit einem anderen Dateisystem verwenden und direkte Anpassungen in der VM müssen nicht

[8] Server Message Block. Kommunikationsprotokoll der Anwendungsschicht des OSI-Models für die Freigabe von Daten und Druckern in Netzwerken.

[9] Ablageorganisation auf einem Datenträger

vorgenommen werden. In der Masterarbeit *Large-Scale, Transparent Format Migration System* [Valizada 2011, Seite 25] der Universität Freibug wird der Umgang mit Containern erklärt und eine Java-Implementierung für I/O-Operationen vorgestellt.

2.5. View-Path

Um mit digitalen Objekten arbeiten zu können, müssen zuerst bestimmte Informationen bekannt sein, diese hierarchisch voneinander abhängen. In Abbildung 2.1 wird dieser Hierarchie als Baum dargestellt. Als erstes muss der Typ des digitalen Objektes festgestellt werden, nur so kann man nach einer Anwendung suchen, mit der das DO manipuliert werden kann.

In vielen Fällen lassen sich mehrere passende Anwendungen finden, die selbst auf verschiedenen Betriebssystemen laufen können. Zuletzt muss man nur wiesen welche Hardware das Betriebssystem fordert, um diese zu emulieren. Auf diese Weise entstehen Verzweigungen im Baum und jeder Weg von der Wurzel bis zu einem Blatt repräsentiert die notwendigen Komponenten für die Manipulation eines bestimmten digitalen Objektes. Einen solchen Weg bezeichnet man als View-Path [Neuroth u. a. 2009, S. VII] und die Komponenten entlang des Pfades (außer der Wurzel) als Sekundärobjekte. In diesem Sinne wird auch die Bezeichnung Primärobjekt für das DO verwendet.

View-Paths haben keine feste Länge, denn je nach Typ des Primärobjektes variiert die Anzahl der benötigten Sekundärobjekte [Philipps 2010, S.14]. So benötigen beispielsweise Primärobjekte wie manche ausführbare Dateien nur zwei Sekundärobjekte, nämlich ein Betriebssystem und dessen geförderte Hardware. Jeder DO-Typ muss wenigstens über einen View-Pfad verfügen, damit dieser in einer emulierten Umgebung manipuliert werden kann [van Diessen und Steenbergen 2002, S.23].

2.6. Assisted Emulation for Legacy Executables

Woods und Brown [2010] beschreiben in der Arbeit Assisted Emulation for Legacy Executables eine Anwendung, um die Notwendigkeit an Vorkenntnisse bei der Benutzung von emulierten Umgebungen zu reduzieren. Sie zeigten mittels AutoIT, eine Software zur Erstellung von Makros für die Automatisierung von Abläufe unter Betriebssystemen der Firma Microsoft, dass es möglich ist, Anwendungen automatisch bei Bedarf zu installieren. So ist es möglich, emulierte Umgebungen nach Anleitung

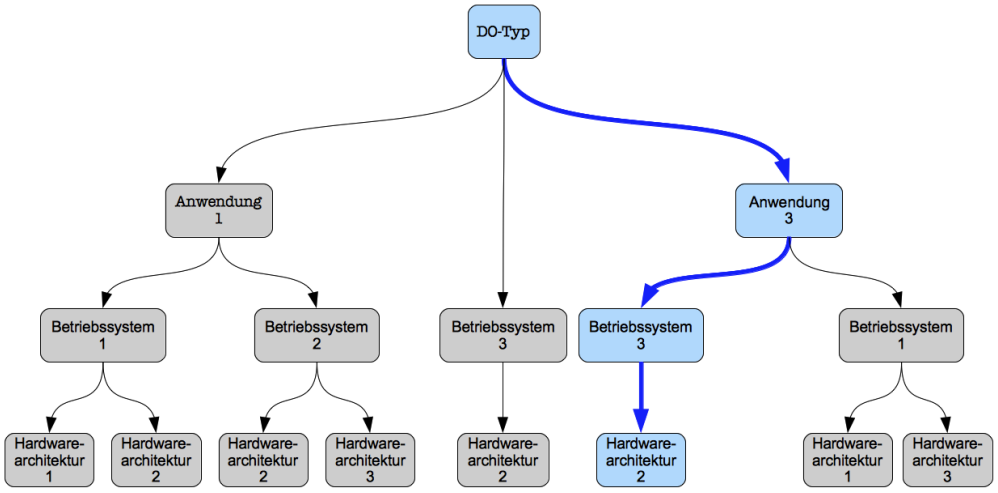

Abbildung 2.1.: View-Path - Beispiel eines hierarchischen Aufbau

eines View-Paths zu erstellen. Mit relativ kleinem Programmieraufwand erstellten sie eine GUI, die der Benutzer sowohl bei der Konfiguration und Ausführung emulierter Umgebungen als auch bei der Ausführung von Installationsskripte assistiert. Dieser Ansatz erfordert sehr gute Kenntnisse über die emulierten Systeme und ist abhängig von der emulierten Hardware-Architektur.

2.7. View-Path Realizations for Obsolete Digital Objects

Ein ähnlicher Ansatz wie der von Kapitel 2.6 wurde von Uhrig [2011] an der Universität Freiburg in Rahmen einer Bachelorarbeit implementiert und getestet. Der Ansatz basiert auf der Ausführung von Skripten wie `AUTOEXEC.BAT` oder dem vorkonfigurierten Autostart von bestimmten Anwendungen innerhalb einer emulierten Umgebung. Die Skripte oder Voreinstellungen haben die Aufgabe, importierte digitale Objekte mit einer geeigneten Anwendung automatisch zu öffnen. Weitere Voraussetzungen sind die Verwendung von vorbereiteten System-Images für unterschiedliche Emulatoren oder Virtualisierungstools so wie ausreichende Informationen über die zu emulierende Umgebung. Dieser Ansatz benötigt keine zusätzlichen Anwendungen und kann auf eine große Menge von Betriebssysteme und Emulatoren angewendet werden. Er wurde mit den Emulatoren DosBox, QEMU und Virtual Box überwiegend bei der Arbeit mit Office Dateien unter DOS getestet.

3. Ansatz

Wie in Kapitel 2.5 erklärt, enthalten View-Paths alle Informationen über die notwendigen Ressourcen für die Erstellung einer emulierten Umgebung, in der ein bestimmter DO-Typ manipuliert werden kann. View-Paths sind der wichtigste Bestandteil des Ansatzes *Assisted Create View*, denn auf ihm basieren alle Operationen, die zur erfolgreichen Darstellung bzw. Ausführung des digitalen Objektes führen. Der Ansatz kann in zwei Phasen, Aufzeichnungs- und Abspielphasen, aufgeteilt werden. Die erste Phase setzt sich damit auseinander, wie die Routine für die Behandlung des Objektes gestaltet wird, damit diese in der zweiten Phase verwendet werden kann.

3.1. Aufzeichnungsphase

In dieser Phase geht es darum, den ganzen Prozess für die Darstellung eines digitalen Objektes aufzuzeichnen mit dem Ziel, diese Routine für Objekte des selben Typs in späteren Zeitpunkten zu benutzen.

3.1.1. Untersuchung des Primärobjektes

Am Anfang ist nur ein digitales Objekt vorhanden. Aus diesem Objekt werden die ersten und relevanten Informationen für die Anwendung des Ansatzes herausgezogen. Je mehr Informationen man aus dem Objekt bekommt, desto präziser wird das Ergebnis des Ansatzes. Da alle weiteren Schritte von diesem DO abhängen, werden wir es ab diesem Moment Primärobjekt nennen.

Anhand der Dateierweiterung des Primärobjektes lässt sich in den meisten Fällen sein Typ feststellen. Digitale Werkzeuge wie PRONOM[1] verfügen über eine große Sammlung an Informationen über Dateitypen, die zu genaueren Untersuchung des Primärobjektes dienen.

[1] Ein Online-System mit Informationen über Dateiformaten. Siehe http://www.nationalarchives.gov.uk/PRONOM/Default.aspx

Fehlt eine Dateierweiterung oder ist sie den zusätzlichen Werkzeugen nicht bekannt, wird anhand des Inhalts des Primärobjektes versucht, den Typ zu erkennen. Eine Erkennung dieser Art ist oft nicht praktikabel auf Grund ihres Aufwands.

Im besten Fall erfährt man den Typ des Primärobjektes und welche Anwendung oder System mit Objekte dieser Art umgehen können. Damit kann man nach bestimmten Hinweisen wie Versionsnummern im Inhalt des Objekts suchen und herausfinden, welche Anwendungs- oder Systemversion am geeignetsten ist. Wenn keine Versionshinweise vorhanden sind, muss eine Entscheidung mittels Statistiken und / oder Benutzerinteraktion getroffen werden. Nach erfolgreichem Abschluss dieser Phase ist die erste Komponente des View-Paths bestimmt. Solche Sekundärobjekte können beispielsweise Anwendungen oder Betriebsysteme sein.

3.1.2. Ressourcen-Sammlung

Mit dem ersten Sekundärobjekt kann man sich auf die Erstellung des vollständigen View-Paths konzentrieren. Hier ist die Größe und Qualität der Information eines vorhandenen Softwarearchivs ausschlaggebend für die Erstellung eines guten View-Paths. Das Softwarearchiv enthält nicht nur Informationen über die Sekundärobjekte sondern auch die Sekundärobjekte selbst, entweder als einzelne Pakete oder innerhalb vorbereiteter Betriebssystem-Images.

Wo und wie die Sekundärobjekte vom Archiv verwaltet[2] werden und die Erzeugungsart[3] der View-Paths, ist an dieser Stelle nicht von besonderer Bedeutung. Wichtig ist, dass das Softwarearchive Anfragen durch die Rückgabe von gültigen View-Paths beantwortet. Ein View-Path ist gültig, wenn alle seine Informationen auf im System vorhandene Sekundärobjekte verweisen. Die Überprüfung der Gültigkeit von View-Paths ist die Aufgabe des Softwarearchiv.

Sobald ein View-Path bestimmt wurde, muss das Archiv ein Betriebssystem-Image anbieten, in dem alle benötigten Sekundärkomponente vorhanden sind. Das Images ist in den meisten Fällen für einen bestimmten Emulator erstellt, so dass dieser als letztes Sekundärobjekt des View-Paths betrachtet werden könnte. Falls weitere im Betriebssystem-Image nicht vorhandene Objekte benötigt werden, müssen diese und die Informationen für ihre Verwendung mitgeliefert werden. Ein typisches Beispiel dafür wäre die Verwendung von Anwendungen, die ihre Benutzung nur in Verbindung mit einer CD erlauben.

[2] [Philipps 2010, Kap. 3.1.2]

[3] Dynamische Erzeugung von View-Paths[Philipps 2010, Kap. 3.2] und Statische Archivierung von View-Paths[Philipps 2010, Kap. 3.3]

3.1.3. Objekt-Transport

Damit ein Objekt in einer emulierten Arbeitsumgebung geöffnet werden kann, muss es sich innerhalb der VM befinden. Durch die Verwendung von Containers ist es möglich, mit relativ wenigen Anforderungen Objekte in eine emulierte Umgebung zu transportieren. Dafür wird ein Image eines Datenträgers erstellt, der das Objekt enthält. Dieses Image wird beim Starten der VM als externes les- und schreibbares Medium angehängt und muss mit einem für die emulierte Umgebung geeigneten Dateisystem formatiert sein. Die Methode ist außerdem eine sehr gute Lösung für Umgebungen ohne Netzwerk-Support.[4]

Bei der Lösung von Problemen mit dem Objekt-Transport müssen keine direkten Änderungen innerhalb der emulierten Umgebung vorgenommen werden, sondern am Datenträger oder am Emulator. Den Datenträger mit einem anderen Dateisystem zu formatieren, den Einhängepunkt und den Typ des Datenträger am Emulator zu ändern, sind einige der typischen Anpassungen.

3.1.4. Emulierte Umgebung starten

An dieser Stelle wird der Emulator mit dem Betriebssystem-Image aus dem Kapitel 3.1.2 als Boot-Medium ausgeführt. Weitere Speichermedien sowie das Image mit dem Primärobjekt werden durch den Emulator als virtuelle Speichermedien im emulierten System eingehängt.

Eine statische Indizierung der Speichermedien und die Bereitstellung einer VNC-Schnittstelle für die Darstellung und Steuerung der emulierten Umgebung sind besondere Voraussetzungen für den nächsten Schritt und können nur durch eine geeignete Konfiguration des verwendeten Emulators erreicht werden. Auf Grund der Abstraktheit des Ansatzes fallen die Ansprüche, die sonst ans Gastsystem gestellt werden, auf den Emulator. Jedoch kann man nicht bei einer Emulator-Konfiguration erwarten, dass alle Gastsysteme gleich darauf reagieren. Die Konfigurationen müssen so allgemein wie möglich sein und nur verträgliche Ansprüche ans Gastsystem stellen.

3.1.5. Routine aufzeichnen und speichern

Routinen für das Öffnen oder Ausführen eines Primärobjektes, die in der Regel vom Benutzer durchgeführt werden und aus Maus- und Tastatur-Events bestehen, werden mittels VNC aufgenommen und an die emulierte Umgebung weitergeleitet. Jede

[4] Vgl. [Valizada 2011, Kap. 2.2]

Aufzeichnung wird als Record-Datei im Softwarearchiv gespeichert und mit den entsprechenden Informationen versehen, damit sie in späteren Zeitpunkten weiter zur Verfügung steht. Routinen können nach Bedarf abgespielt werden und ermöglichen somit die Automatisierung bestimmter Aktionen.

Obwohl die Record-Dateien später von Software-Archiv angeboten werden, sind sie keine Sekundärobjekte, denn sie sind nicht entscheidend für die Darstellung oder Ausführung des Primärobjektes. Die Record-Dateien sind hier nur Hilfsobjekte für Automatisierungen.

Die Verwendung von VNC bringt viele Vorteile mit sich und ist ein wesentlicher Bestandteil des *Assisted-Create-View*-Konzepts. Durch VNC funktioniert die Steuerung in der emulierten Umgebung unabhängig von den installierten Eingabemedien und erlöst von der Aufgabe, in der Emulation einzudringen um Maus- und Tastatur-Events auszulösen. Auf Grund der Einfachheit des Verfahrens können Routinen vom Verwalter des Softwarearchivs erstellt und gespeichert werden ohne Programmierkenntnisse oder komplexe Überlegungen.

An dieser Stelle muss erwähnt werden, dass solche Vorteile nicht nur durch den Einsatz von VNC erreicht werden können. Es gibt eine relativ große Auswahl an Lösungen wie Citrix XenApp[5] und FreeNX[6], nur um ein paar Beispiele zu nennen, die auf Protokollen wie RDP[7], NX[8] und ICA[9] basieren[10]. Nichtsdestotrotz zeigen Arbeiten wie die von Tchayep [2011], Kulzhabayev [2012] und Philipps [2010], dass VNC ein gut getestetes Tool ist und auf Grund seines Entwicklungsgrades sich bisher annähernd als beste Lösung hervorgetan hat.

3.2. Abspielphase

Hier so wie in der Aufzeichnungsphase aus Kapitel 3.1 wird nach Sekundärobjekten für ein Primärobjekt gesucht. Die Existenz einer Record-Datei assoziiert mit dem benötigten View-Path im Softwarearchiv wird vorausgesetzt, damit das Öffnen

[5] Remote-Desktop-Software der Firma Citrix. Siehe http://www.citrix.com
[6] Remote-Desktop-Software von BerliOS. Siehe http://freenx.berlios.de
[7] Proprietäres Netzwerkprotokoll von Microsoft. Siehe http://url9.de/ivZ
[8] Remote-Desktop-Software des italienischen Unternehmens NoMachine. Siehe http://www.nomachine.com
[9] Plattformunabhängiges proprietäres Protokoll der Firma Citrix Systems.
[10] Für einen ausführlichen Vergleich zwischen den meist verwendeten Remote-Desktop-Anwendungen siehe http://en.wikipedia.org/wiki/Comparison_of_remote_desktop_software

oder Ausführen des Primärobjektes in der emulierten Umgebung automatisiert werden kann. Am Ende dieser Phase soll der Benutzer zu einer emulierten Umgebung gelangen, in dem das Primärobjekt bereits ausgeführt oder mit einer geeigneten Anwendung geöffnet wurde.

Wie in der Kapitel 3.1 wird hier als erstes versucht, den Typ des digitalen Objektes festzustellen. In erfolgreichem Fall wird der DO-Typ erkannt und eine Suche im Softwarearchiv ergibt mindestens einen gültigen View-Path und eine Record-Datei. Das DO wird wie in Kapitel 3.1.3 in einem geeigneten digitalen Container verkapselt und einem Emulator zur Verfügung gestellt. Das starten der emulierten Umgebung geschieht wie im Kapitel 3.1.4, allerdings stellt hier das Emulator zusätzlich zur VNC-Schnittstelle eine abstrakte SDL[11]-Schnittstelle für Grafik-, Sound- und Eingabegeräte bereit. Mittels dieser Schnittstelle soll der Benutzer die emulierte Umgebung am Ende der Automatisierung steuern.

Sobald die emulierte Umgebung hochgefahren wurde, wird die aufgezeichnete Routine aus dem Record-Datei abgespielt. Alle Maus- und Tastatureingaben werden gelesen und über die vom Emulator bereitgestellte VNC-Schnittstelle ans Gastsystem weitergegeben. Für eine gewisse Gewährleistung der Sicherheit kann die kurze kryptologische Prüfsumme *Fingerprint* zum Einsatz kommen. Die Abbildung 3.1 zeigt die Struktur des Systems für das Aufnehmen (capture) und Abspielen (playback) von Routinen mittels VNC.

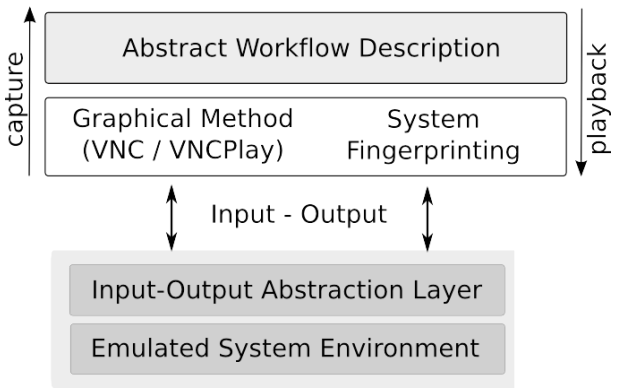

Abbildung 3.1.: Funktionsweise der Aufzeichnung von Benutzerinteraktionen [Rechert u. a. 2009, S.4]

Wurde das Primärobjekt erfolgreich geöffnet oder ausgeführt, dann darf der Benutzer die Kontrolle des emulierten Umgebung übernehmen. Da man die VNC-

[11] Simple DirectMedia Layer. Eine freie Multimedia-Bibliothek für verschiedene Plattformen. Siehe http://www.libsdl.org

Schnittstelle nicht mehr benötigt, wird diese beendet. So bleibt nur die SDL-Schnittstelle offnen, durch die sich die emulierte Umgebung sich vom Benutzer steuern lässt.

3.2.1. Probleme

Oft treten Fehler beim Abspielen der aufgezeichneten Routinen auf wegen Versatz des Mauszeigers[12]. Gründe dafür sind die Verwendung von unterschiedlichen Betriebssystemen im Host- und im Gastsystem so wie die Unterschiede zwischen der emulierten und der echten Hardware. Kulzhabayev [2012] testete und analysierte in Rahmen seiner Master-Arbeit den Arbeitsablauf beim Abspielen von gespeicherten Routinen in emulierten Umgebungen über VNC und sortierte die Gründe des Versatzes in folgenden Gruppen:

1. Effect of Time Interval of Event Generation: Je nach Duration des Events regiert der Mauszeiger im Gastsystem anders.
2. Effect of Processor Load: Die Zeit, die dei CPU benötigt um ein empfangenes Event auszulösen verursacht eine Verzögerung der gesamten Folge von Events.
3. Effect of Disk I/O Load: Die Ausführung von vielen I/O-Operationen im Gastsystem beim Abspielen einer Folge von Events führt zu fehlerhaften Ergebnissen.
4. Effect of Pointer Acceleration: Versatz des Mauszeigers auf Grund von Einstellungsunterschieden bei der Beschleunigungen der Mauszeiger im Host- und im Gastsystem.

3.3. Diskussion

Ein wichtiger Aspekt für die Implementierung des *Assisted-Create-View*-Ansatzes ist die Methode, die für die Übergabe der vorbereiteten emulierten Umgebung an den Benutzer verwendet wird.

Für das Testen und die Analyse verschiedener Methoden werden der Emulator QEMU und der JAVA VNC Client VNCplay[13] verwendet. Die wichtigsten Anforderungen sind:

[12] Horizontale und vertikale Abweichung von der Standardposition des Mauszeiger auf dem Bildschirm.

[13] Anwendung für das Aufzeichnen und Abspielen von VNC-Sitzungen entwickelt an der Stanford University. Siehe Nickolai Zeldovich [2005]

1. Der Benutzer darf nicht beim Abspielen einer Routine eingreifen.

2. Bei der Übergabe der Umgebung soll weder die Performance noch die Stabilität des Systems benachteiligt werden.

3. Die Schnittstelle, durch die der Benutzer die emulierte Umgebung steuern soll, muss für ihn so uneingeschränkt und bekannt wie möglich sein.

3.3.1. Nur VNC

In dieser Variante werden nur die VNC-Schnittstelle und VNCplay benutzt. VNC wird sowohl für das automatisierte Öffnen des Objektes in der emulierten Umgebung als auch als Steuerungssystem der Umgebung für den Benutzer verwendet.

Vorteile

Die Verwendung von VNC sowohl für die Steuerung als auch für die Darstellung der emulierten Umgebung schafft eine gewisse Einheitlichkeit im Konzept und trägt dazu bei, dass die Anwendung schneller und stabil läuft.

Nachteile

Der Gebrauch von VNC als Hauptsteuerungsprotokoll emulierter Umgebungen ist nur in einer eingeschränkten Weise dienlich. So können beispielsweise nicht alle Tastenkombinationen verwendet werden. Außerdem ist auch mit Verzögerungen zu rechnen. Dies stellt ein Problem dar besonderes bei Anwendungen, die eine schnelle Reaktion des Benutzers verlangen.

3.3.2. VNC + Suspend + SDL

Damit die Automatisierung für das Öffnen des Objektes durchgeführt werden kann, wird zuerst der Emulator mit einer VNC-Schnittstelle gestartet und VNCplay spielt die Aufzeichnung aus einer Record-Datei ab. So bald die Automatisierung vollständig ausgeführt wurde, wird die emulierte Umgebung suspendiert und gleich wieder

gestartet, allerdings nun mit einer SDL-Schnittstelle, so dass der Benutzer in der vorbereiteten Umgebung mit dem Editieren bzw. der Ausführung des Objektes anfangen kann.

Den aktuellen Zustand der emulierten Umgebung speichert man mit dem Befehl

```
(qemu) savevm ZUSTANDSNAME
```

in der QEMU-Konsole. Diese Operation nimmt einige Minuten im Anspruch. Falls das System im Snapshot-Modus betrieben wird, bleibt dieser Zustand gespeichert nur bis QEMU endet, ansonsten bleibt der Zustand auf der virtuellen Boot-Platte gespeichert und kann mit dem Befehl

```
(qemu) loadvm ZUSTANDSNAME
```

in der QEMU-Konsole oder beim Starten von QEMU mit dem Parameter

```
-loadvm ZUSTANDSNAME
```

wiederhergestellt werden. Sowie das Speichern nimmt auch das Laden einige Minuten im Anspruch.

Die Befehle

```
(qemu) info snapshots
(qemu) delvm ZUSTANDSNAME
```

in der QEMU-Konsole können jeweils gespeicherte Zustände auflisten und löschen.

Vorteile

Beim Suspendieren der emulierten Umgebung wird automatisch eine Art Kontrollpunkt erstellt. Dieser Kontrollpunkt kann als Zustand vom Ablauf des im Kapitel 3.1 beschriebenen Verfahrens betrachtet werden. Falls währen des o.g. Prozesses ein Fehler auftritt, kann der Kontrollpunkt wiederhergestellt werden und das Prozess ab diesem Punkt weiter ausführen anstatt wieder von Vorne anzufangen. Außerdem schließt man den Nachteil vom Ansatz VNC + SDL aus, da keine zusätzliche graphische Oberflächen gleichzeitig offen sein müssen.

Nachteile

Das Suspendieren der emulierten Umgebung bringt in der Praxis eine gewisse Instabilität mit sich. Jedes mal wenn eine VM suspendiert wird, wird ein Snapshot[14] angelegt. In diesem Snapshot befindet sich sowohl der aktuelle Zustand der Umgebung als auch eine Kopie des vom Emulator benutzten Hauptspeichers. Beim Aufwecken einer suspendierten Umgebung , werden alle Daten aus dem Snapshot wiederhergestellt und so indiziert, dass die Umgebung selbst nichts davon mitbekommt und ganz normal weiter arbeitet. Der geringste Fehler bei diesem Verfahren kann die emulierte Umgebung zum Abstürzt bringen, außerdem nimmt es sehr viel Zeit in Anspruch, so dass dieser Ansatz unter Umständen nicht praktikabel ist.

3.3.3. VNC + SDL

Der Emulator startet gleichzeitig mit den Schnittstellen SDL und VNC.[15] Die Automatisierung für das Öffnen des Digitalen Objekts in der emulierten Umgebung erfolgt über die VNC-Schnittstelle und VNCplay, welche am Ende der Automatisierung beendet werden. So bleibt nur die SDL-Schnittstelle offen, durch die der Benutzer die Umgebung selbständig steuern kann.

Um QEMU mit den Schnittstellen SDL und dem VNC gleichzeitig starten zu können, sind folgende Parameter zu Verwenden:

```
-sdl -vnc :X
```

So bald die VNC-Schnittstelle nicht mehr benötigt wird, schließt man sie mit dem Befehl

```
(qemu) change vnc none
```

in der QEMU-Konsole. Obwohl der VNC-Server für neue Verbindungen nicht mehr zu Verfügung steht, werden bestehende Verbindungen aufrecht erhalten.

Vorteile

Durch die parallele Ausführung von zwei grafischen Schnittstellen ist das Suspendieren der emulierten Umgebung nicht notwendig.

[14] Aufzeichnung des globalen Zustandes einer emulierten Umgebung zu einem bestimmten Zeitpunkt
[15] Für diese Methode wird die Verwendung der QEMU-Version 0.14.0 (qemu-kvm-0.14.0) oder höher vorausgesetzt.

Nachteile

Zwei gleichzeitig geöffnet grafische Schnittstellen könnten den Benutzer verwirren.

3.3.4. Schlussfolgerung

Nach etlichen Tests stellte sich heraus, dass VNC für die Hauptsteuerung emulierter Umgebungen nicht geeignet ist. Gründe dafür sind insbesondere Verzögerungen von Maus- und Tastatureingaben sowie die Einschränkung der Tastenkombinationen. Die meisten Probleme traten auf bei der Ausführung von Anwendungen mit viel Benutzer-Interaktion. Spiele, Bildbearbeitungsprogramme oder Simulationsprogramme sind gute Beispiele, bei denen die Verwendung von VNC keine sinnvolle Lösung ist, denn hier sind Reaktionszeit, Genauigkeit bei der Positionierung des Maus-Zeigers und die Verwendung von vielen Tastenkombinationen notwendig. Nichtsdestotrotz gab es bei der Arbeit mit Office-Anwendungen wie Kalkulations-, Textbearbeitungsprogrammen u.ä. keine großen Schwierigkeiten.

Die Methode *VNC + Suspend + SDL* schien am Anfang die allerbeste Lösung zu sein. So wie bei der Methode *nur VNC* ist es hier auch durch die Verwendung von VNCplay nicht möglich, dass der Benutzer während des Abspielens einer Routine eingreift. Die SDL-Schnittstelle bietet dem Benutzer eine relativ uneingeschränkte und angenehme Steuerung der Umgebung ohne Verzögerungen bei der Benutzung von Eingabegeräten. Allerdings nimmt in der Tat das Suspendieren und Aufwecken der Umgebungen so viel Zeit in Anspruch, dass diese Lösung nicht praktikabel ist. Auch die Stabilität der Umgebungen wird beim Aufwecken beeinträchtigt Besonderes von diesem Nachteil betroffen sind Emulationen von alten Betriebssystemen und Emulationen mit Betriebssystem-Images von großem Umfang.

Gute Stabilität und Performance des Systems machen aus der Methode *VNC + DSL* die beste Lösung von allen. Auch hier wird die atomare Ausführung von Routinen durch VNCplay gewährleistet und in Laufe der Test-Phase erwies sich die SDL-Schnittstelle für die Steuerung der Umgebung nach dem Automatisierungsprozess als schnell und präzise. Aus diesen Gründen wird diese Methode bei der Implementierung des Ansatzes angewendet.

4. Implementierung

Um den Ansatz *Assisted Create View for Digital Artefacts* in der Praxis testen zu können, wurde die Implementierung acv4ad[1] entwickelt. Die in JAVA geschriebene Anwendung basiert auf der Anwendung VNCplay von Nickolai Zeldovich [2005] und deren Erweiterung von Ruzzoli [2010]. VNCplay kann interaktiver VNC-Sitzungen aufnehmen und abspielen, außerdem ermöglicht ihre zustandsbasierte Arbeitsweise die Erkennung und Behandlung von Fehlern[2].

Für die Emulation verwendet ACV4DA den Emulator QEMU, welcher durch einen eingebauten VNC-Server Zugriff auf das emulierte Gastsystem erlaubt. Da der VNC-Server im Emulator und nicht im Gastsystem läuft, bleibt der Ansatz nach wie vor völlig abstrakt. In Abbildung 4.1 wird die Struktur der Interaktion zwischen VNC und QEMU präsentiert. Da die Emulation sich im lokalen Rechner befindet, wird die Loopback-Schnittstelle[3] verwendet, um ein Netzwerk zu simulieren.

Speicherung von View-Paths und Record-Dateien

Maus- und Tastatureingaben, die über VNC an den Emulator weitergegeben werden, werden in sogenannten Record-Dateien gespeichert und mit View-Paths assoziiert. Die Speicherung der View-Paths und ihre Assoziierung mit Record-Dateien erfolgt durch die Verwendung von XML-Dateien, die die Aufgabe haben, die Interaktion mit einem Software-Archiv zu simulieren.

Pro DO-Typ gibt es eine XML-Datei, in der die notwendigen Ressourcen für die Verwendung digitaler Objekte dieses Typs aufgelistet sind. Sind die notwendige Ressourcen für die Verwendung eines digitalen Objekts bekannt, so kann man nach einer Umgebung suchen, in der diese Ressourcen vorhanden sind. Die zu emulierenden

[1] Demonstration and Evaluation of an Assisted Create View for Digital Artefacts on an Emulation System Workstation. Siehe http://projects.jcgt.de/projects/acv4da

[2] Siehe [Ruzzoli 2010, Kap. 3.2]

[3] Über die Loopback-Schnittstelle werden Pakete geroutet, die zwischen zwei Prozesse auf demselben Rechner ausgetauscht werden. Siehe RFC 3330 und RFC 4291.

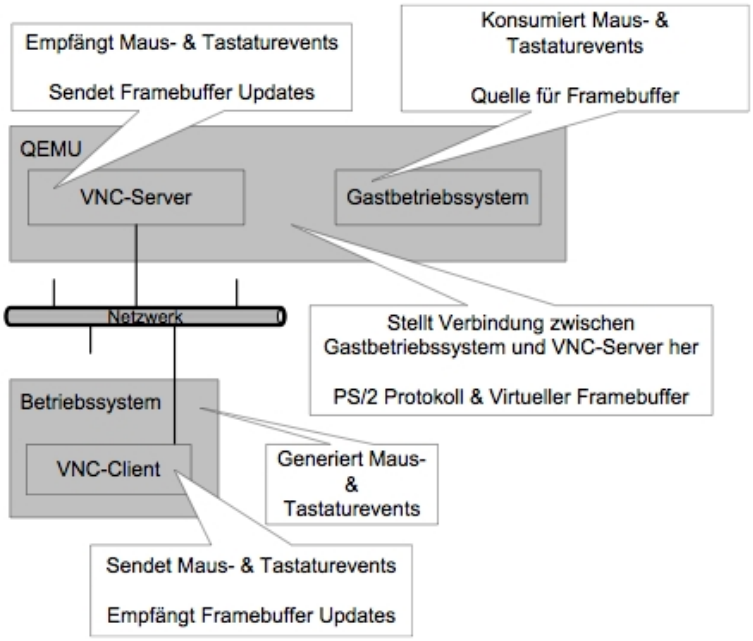

Abbildung 4.1.: Das RFB Protokoll als Schnittstelle zur virtuellen Maschine [Philipps 2010, S. 6]

Umgebungen werden in Form von virtuellen Datenträgern gespeichert. Jeder virtuelle Datenträger ist die Abbildung einer Festplatte, auf der eine bestimmte Software-Sammlung vorinstalliert ist. Um die Suche nach Ressourcen innerhalb eines virtuellen Datenträgers zu ermöglichen, werden alle Informationen über jeden dieser Datenträger in einer getrennten XML-Datei gespeichert.

Da die Aufzeichnungen für die Automatisierung von Öffnungs- bzw. Ausführungsvorgängen in Abhängigkeit mit einer Umgebung stehen, wird hier außerdem aufgelistet, welche Record-Dateien unter dieser Umgebung verwendbar sind. Sobald die Anwendung festgestellt hat, welcher View-Path für ein Objekt zu verwenden ist, startet sie die passende emulierte Umgebung und spielt am Ende dieses Vorgangs die Routine aus der assoziierten Record-Datei ab.

Aufzeichnen von Maus- und Tastatur-Eingaben

Für den Fall, dass für das Öffnen oder Ausführen eines digitalen Objektes keine Record-Datei vorhanden ist, bietet die Anwendung die Möglichkeit, eine Record-Datei zu erstellen und zu assoziieren. So bald ACV4DA die Emulation startete, muss

der Benutzer, wie in Abbildung 4.2 gezeigt, die Aufzeichnung initialisieren und den Prozesse für das Öffnen oder Ausführen des digitalen Objekts durchführen.

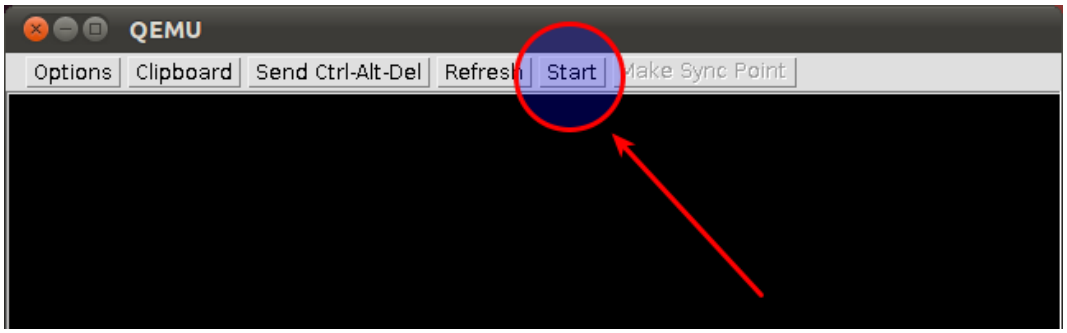

Abbildung 4.2.: Initialisierung des Aufnahme-Vorgangs

Ist dieser Vorgang abgeschlossen, dann muss man nur, wie in Abbildung 4.3 gezeigt, die Aufzeichnung pausieren und das Fenster schließen. Die Emulation wird neugestartet und die gespeicherte Aufzeichnung abgespielt, um zu überprüfen, ob die Aufzeichnung richtig funktioniert. Falls diese fehlerhaft ist, kann man den Vorgang wiederholen. Eine Aufzeichnung wird endgültig gespeichert bis diese vom Benutzer durch ein Dialogfenster als verwendbar gekennzeichnet wird.

Benutzerinteraktion

Unter Umständen können auch mehrere View-Paths vorhanden sein und der Benutzer muss sich für eine der angebotenen Varianten entscheiden. Alle verwendbare View-Paths werden in einer GUI aufgelistet, um den Auswahlvorgang intuitiver zu gestalten. Die in Abbildung 4.4 präsentierte GUI wurde in der Klasse `ACV4DAProfile-Editor` implementiert und kümmert sich um die endgültige Sammlung von Ressourcen. Sobald der Benutzer sich für einen View-Path entschieden hat, startet die Klasse die Emulation mit allen gesammelten Ressourcen.

Ein gesamter Überblick über die Arbeitsweise der Anwendung und wie diese mit dem Software-Archiv interagieren soll, wird in Abbildung 4.5 vorgestellt. Die Kommunikation zwischen der Anwendung und dem Softwarearchiv wird durch eine abstrakte Klasse namens `ACV4DACommunicator` simuliert. Diese Klasse sucht in den XML-Dateien nach gültigen View-Paths und ermittelt diese als Zeichenketten. Die Zeichenketten enthalten Pfade zu vorhandenen Ressourcen im System, die für die Erstellung der emulierten Umgebung notwendig sind.

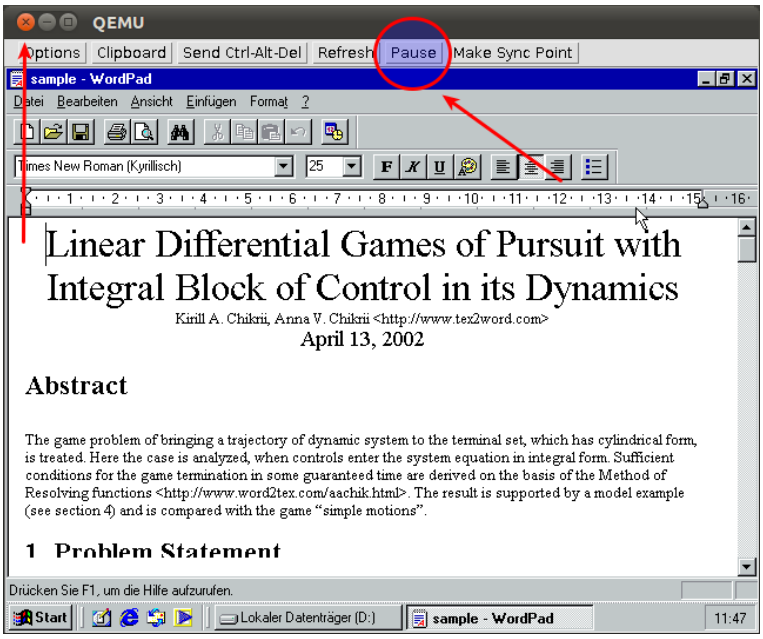

Abbildung 4.3.: Ende des Aufnahme-Vorgangs

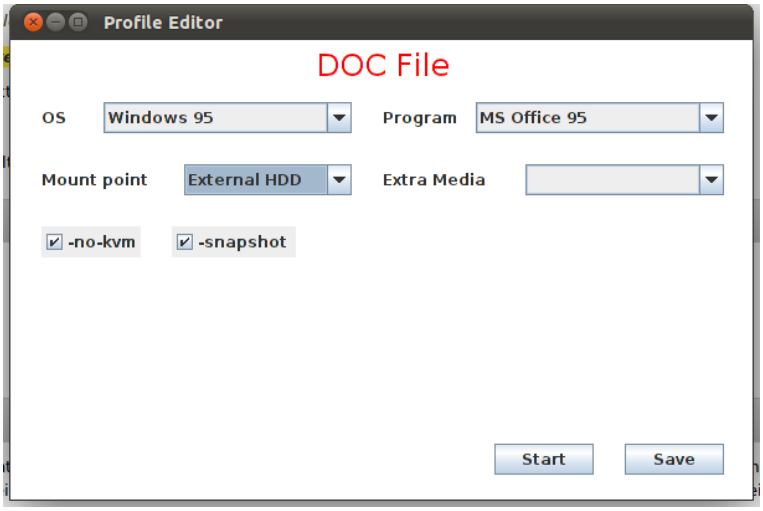

Abbildung 4.4.: GUI für Auswahl eines View-Paths

Bei diesen Ressourcen handelt es sich um die im Kapitel 3.1.2 genannte Sekundärobjekte. Die essentiellen Ressourcen für die Erstellung einer emulierten Umgebung sind:

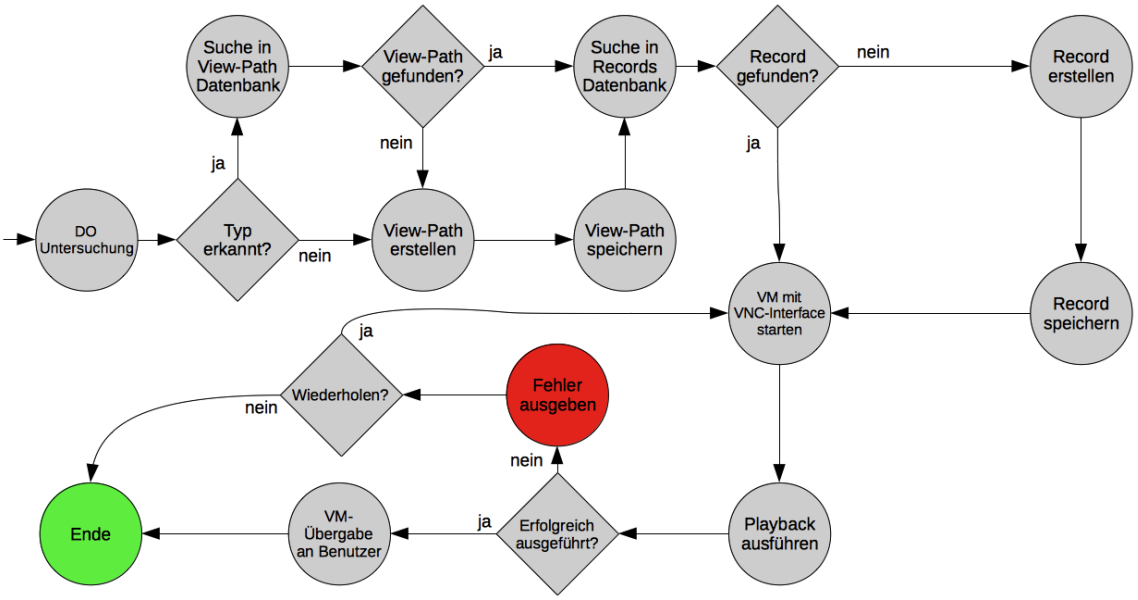

Abbildung 4.5.: Assisted Create View for digital Artefacts

1. Ein Emulator, der die benötigte Rechnerarchitektur emulieren kann.
2. Ein Betriebssystem-Image, dass die notwendigen Sekundärobjekte enthält und mit dem o.g. Emulator kompatibel ist.
3. Eine Record-Datei damit das Öffnen oder das Ausführen des Objektes in der emulierten Umgebung automatisiert werden kann.

Je nach Art der Sekundärobjekte könnte auch eine vierte Komponente, das Image eines Datenträgers, notwendig sein. Dieses Image muss als virtueller Datenträger in der Umgebung eingehängt werden. Zusätzlich dazu wird auch die Information geliefert, wie dieses Image zum einhängen ist. Unterstütze Varianten sind Floppy, CD oder DVD.

Ein Beispiel für die Ausführung des Emulators QEMU mit allen gewöhnlichen notwendige Komponenten sieht folgendermaßen aus:

```
qemu -drive file=data/drivers/win95.img,snapshot=on\
-drive file=data/drivers/hdd.img,snapshot=off\
-sdl -vnc :0
```

Für den Transport von digitalen Objekten in die Emulation werden die Skripts `hdd_create.sh`[4] und `hdd_io.sh`[5] von Valizada [2011] verwendet. Das Skript `hdd_create.sh` erstellt die Abbildung eines virtuellen Datenträger, der vom Emulator als Wechseldatenträger in der emulierten Umgebung eingehängt wird. Das Skript `hdd_io.sh` kümmert sich vor der Emulation um das Hinzufügen der digitalen Objekte in den virtuellen Datenträger und nach der Emulation um das Extrahieren dieser Objekte. Alle extrahierte Objekte werden in den Ordner `/data/extraxt/` gespeichert.

Zeitlimitierte Ausführungen

Um zu Verhindern, dass die Anwendung durch unerwartete Verhalten von Subprozesse hängen bleibt, kann die maximale Ausführungszeit jeder Prozess eingestellt werden. Überschreitet ein Subprozess seine maximale Ausführungszeit, so wird dieser und somit auch der Hauptprozess beendet. Nach dem Abbruch wird der Benutzer darüber informiert, an welcher Stelle die Anwendung gescheitert ist.

[4] Siehe http://lab.ks.uni-freiburg.de/projects/digital-preservation/wiki/HDD_Creation_Script

[5] Siehe http://lab.ks.uni-freiburg.de/projects/digital-preservation/wiki/HDD_InputOutput_Script

5. Analyse und Ergebnisse

Anhand der Anwendung ACV4DA wurde eine Reihe von Experimenten durchgeführt, um die Zuverlässigkeit des Ansatzes *Assisted Create View* zu testen. Jedes Experiment besteht aus der Reproduktion einer aufgezeichneten Routine für das Öffnen oder Ausführen eines digitalen Objektes in einer emulierten Umgebung. Die Routinen bestehen in der Regel beinahe nur aus Mauseingaben, dennoch können viele von denen durch Tastatureingaben ersetzt werden. Für die Experimente wurden 15 verschiedene Routinen aufgezeichnet. Danach wurden dieselben Routinen wieder aufgezeichnet nur diesmal wurde auf die Mauseingaben so weit es ging verzichtet, so sind weitere 15 Routinen entstanden.

Die folgende Tabelle enthält die Ergebnisse aus einer Reihe von 150 Versuchen mit den ersten 15 Routinen. Alle eingetragene Werte beziehen sich ausschließlich auf die fehlgeschlagenen Ausführungen. Die Quote von Fehlversuchen liegt bei 18,66% und die Erfolgsquote bei 81,33%. Die Ursachen der meisten Fehler sind Synchronisationsstörungen bei der Übertragung von Maus- und Tastatureingaben durch VNC. Bei diesem Versuch lag der durchschnittliche Anteil an Maus-Interaktion bei ca. 90% und der durchschnittliche Anteil an Tastatur-Interaktion bei ca. 10%.

No.	**Grund des Fehlers**	**Häufigkeit**	**Anteil**
1.	Zeit-Synchronization	10	6,66%
2.	Maus-Verschiebung	10	6,66%
3.	Unerwartetes Dialogfenster	3	2%
4.	Unbekannt	3	2%
5.	Virtuelle Wechseldatenträger nicht eingehängt	2	1,33%
	Gesamte Anzahl von Fehlern	28	18,66%

Eine weitere Versuchsreihe wurde mit der zweiten Gruppe von Routinen durchgeführt. Diese hatten einen durchschnittlichen Anteil an Maus- und Tastatur-Interaktion von jeweils ca. 20% und 80%. Das Experiment wies eine erstaunliche Verminderung der Anzahl von Fehlversuchen auf. Die Quote von Fehlversuchen lag bei 8% und die Erfolgsquote bei 92%.

No.	Grund des Fehlers	Häufigkeit	Anteil
1.	Zeit-Synchronization	2	1,33%
2.	Maus-Verschiebung	3	2%
3.	Unerwartetes Dialogfenster	3	2%
4.	Unbekannt	2	1.33%
5.	Virtuelle Wechseldatenträger nicht eingehängt	2	1,33%
	Gesamte Anzahl von Fehlern	12	8%

Obwohl der Anteil an Fehlversuchen bei den ersten 150 Versuchen relativ hoch ist, muss man berücksichtigen, dass es sich bei ACV4DA um eine Test-Anwendung in der Beta-Phase handelt. Eine ausgereifte Anwendung kann mit Sicherheit die Synchronisationsmängel deutlich reduzieren und somit aus den Ansatz *Assisted Create View* eine praktikable Lösung machen.

5.1. Assisted Create View vs. View-Path Realizations

Während beim Ansatz *Assisted Create View* die graphisch basierte Automatisierung der Öffnungs- oder Ausführungsvorgänge digitaler Objekten in emulierten Umgebungen keine direkte Änderungen innerhalb der Emulationen impliziert, gibt es andere Verfahren, wie der Ansatz von Uhrig [2011], die Automatisierungen durch die Ausführung von Skripts innerhalb der Emulation erledigen.

In der Arbeit *View-Path Realizations for obsolete digital Objects*[1] wird ausführlich beschrieben wie mit Hilfe von Containers, Emulatoren und Skripts emulierte Umgebungen nach Anforderungen eines digitalen Objektes automatisiert angefertigt werden können. Die Containers enthalten die notwendigen Sekundärobjekte und die Skripts eine Reihe von Befehlen, die gezielt nacheinander ausgeführt werden. Diese beide Komponente befinden sich in der emulierten Umgebung und kommen auch dort zum Einsatz.

Die meisten Vorteile der Ausführung von Unterprogramme innerhalb einer Emulation gegenüber des Ansatzes *Assisted Create View* sind besonders bei der Ausführung von Anwendungen und der Assoziation zwischen Anwendungen und Objekten zu finden. Werden Befehle durch Skripts ans System weitergereicht, so empfängt es die Befehle und bearbeitet sie sobald Ressourcen vorhanden sind. Dagegen setzt die Auslösung von Ereignisse durch grafische Oberflächen voraus, dass die entsprechenden Oberflächen über genügend Ressourcen verfügen und bereit sind, Ereignisse zu empfangen um dadurch andere Ereignisse auszulösen.

[1] Uhrig [2011]

Mittels graphischer Oberflächen kann ein Objekt mit einer in der emulierten Umgebung installierten Anwendung auf zwei Weisen geöffnet werden:

1. Durch die Navigation in einem Anwendungsmenü oder Ordnern wird die Anwendung lokalisiert und durch Maus- oder Tastatureingaben ausgeführt. Innerhalb der Anwendung wird durch ein Navigationssystem nach dem gewünschten Objekt gesucht, um dieses letztendlich zu öffnen.

2. Durch die Navigation in Ordnern wird das Objekt lokalisiert und durch Maus- oder Tastatureingaben mit der gewünschten Anwendung geöffnet. Dies setzt natürlich voraus, das die emulierte Umgebung bereits über eine Assoziation zwischen Anwendung und Objekt verfügt.

Durch die Verwendung von Befehlen dagegen, können Objekte in der Regel mittels folgender Syntax mit einer bestimmten Anwendungen geöffnet werden.

```
ANWENDUNGSNAME DATEI
```

Für die Automatisierung von Ereignisse ist der Ansatz mit Unterprogrammen innerhalb emulierter Systeme sehr weit fortgeschritten. Nichtsdestotrotz gibt es Anwendungsbereiche, in denen der Ansatz *Assisted Create View* unvermeidlich zum Einsatz kommt. Solche Fälle treten auf bei der Verwendung von Anwendungen, die extra für Touchscreens entwickelt wurden und sich nicht über Skripts steuern lassen oder nur hauptsächlich durch *Drag and Drop*[2]-Aktionen gesteuert werden können.

Später wenn die heutige Anwendungen für Pads und ähnliche Geräte archiviert werden müssen, wird *Assisted Create View* eine besondere Rolle spielen.

[2]Eine Methode zur Bedienung grafischer Benutzeroberflächen von Rechnern durch das Bewegen grafischer Elemente mittels eines Zeigegerätes

5. Analyse und Ergebnisse

6. Fazit

Die Erschaffung einer Abstraktionsschicht für die Steuerung emulierter Umgebungen ist in der Tat eine gute generische Lösung für Umgebungen, die sich über Maus und Tastatur steuern lassen. Der Ansatz ist nicht nur für ältere und aktuelle Systeme nützlich, er legt auch die Grundlagen für die Arbeit mit zukünftigen Technologien, was ein wichtiger Aspekt der funktionalen Langzeitarchivierung ist.

Es ist mit Verzögerungen und unerwarteten Reaktionen bei der Übertragung von Maus- und Tastatureingaben ins Gastsystem zu rechnen, die allerdings durch Einstellungsanpassungen im Gastsystem deutlich reduziert werden können. Der größte Nachteil des Ansatzes *Assisted Create View* ist seine statische Arbeitsweise, bei der eine entsprechende Reaktion auf Änderungen in den automatisierten Arbeitsabläufen sehr eingeschränkt ist. Zusätzliche Features wie die Verwendung von Synchronisationspunkten bei der Aufzeichnung und Reproduktion von Aktivitäten der Eingabemedien machen den Ansatz zustandsbasiert und schaffen somit eine Pseudodynamik in dem sonst völlig statischen Verfahren.

Der Ansatz *Assisted Create View* hat viel Potential und viele Anwendungsbereiche. Seine Unabhängigkeit von der gesteuerten Umgebung und seine einfache und intuitive Wirkungsweise zeichnen ihn aus. Eine Weiterentwicklung in diesem Bereich könnte zuverlässige und stabile Anwendungen liefern, die mit ähnlichen Verfahren bestimmte Zustände eines Systems erkennen und automatisierte Routinen nach Bedarf ausführen. Die Macht dieses Ansatzes liegt eher in der Geschicklichkeit seiner Implementierung als in seiner Komplexität.

6. Fazit

Anhang

A. Einrichtung der Test-Umgebung mit ACV4DA

Betriebssystem

Ubuntu 11.10 (Getestete Version)[1]

Abhängigkeiten

Folgende Abhängigkeiten müssen im System vorhanden sein:

```
1 qemu (Version 0.14.0 oder hoeher)
2 kvm-pxe
3 sfdisk
4 mkfs
5 dd
6 losetup
7 sync
8 Java Development Kit (Version 6 oder hoeher)
```

Code herunterladen

```
git clone planets@132.230.4.14:~/repositories/vncplay/vncplay_javier.git
```

Anpassungen im System

Für die Verwendung der Skripts `hdd_create.sh` und `hdd_io.sh` muss der Benutzer Zugriff auf die Loop-Devices in `/dev/loop` haben. In der Regel muss nur der Benutzer in die Gruppe `disk` eingefügt werden.

[1] Siehe `http://releases.ubuntu.com/`

Arbeitsumgebung einrichten

- Installieren Sie Eclipse SDK[2]
- Importieren Sie den Code unter
  `File` ⟶ `Import` ⟶ `Existing Projects into Workspace` ⟶ `Select root directory` ⟶ Pfad zum `vncplay_javier` eingeben ⟶ `Assisted Create View for Digital Artefacts` auswählen ⟶ `Finish`

[2]Getestete Version: 3.5.2. Siehe `http://www.eclipse.org/`

Literaturverzeichnis

ALUF 2006
ALUF: First Version of GRATE / PLANETS. URL http://www.planets-project.eu/docs/reports/Planets_PA5-D7_GRATE.pdf, 2006 (IST-2006-033789). – Forschungsbericht

van Diessen und Steenbergen 2002
Diessen, Dr. Raymond J. van ; Steenbergen, Drs. Johan F.: *The Long-Term Preservation Study of the DNEP project - an overview of the results*. IBM Netherlands, Amsterdam, 2002. – ISBN 90-6259-154-X

Drümmer u. a. 2007
Drümmer, O. ; Oettler, A. ; Seggern, D.: *PDF/A kompakt: digitale Langzeitarchivierung mit PDF*. Callas Software, 2007. – ISBN 9783981164800

Höfler 2008
Höfler, B.: *Preserving Digital Media- Hardware: Digitale Langzeitspeicher*. GRIN Verlag GmbH, 2008. – URL http://books.google.de/books?id=eENsZBCWh_8C. – ISBN 9783638883146

Kulzhabayev 2012
Kulzhabayev, Alibek: *Abstract Unattended Workflow Interactions*. January 2012

Neuroth u. a. 2009
Neuroth, H. ; Osswald, A. ; Scheffel, R. ; Strathmann, S. ; Jehn, M.: *Eine kleine Enzyklopädie der digitalen Langzeitarchivierung*. 2. Verlag Werner Hülsbusch, Boizenburg, 2009. – URL http://nbn-resolving.de/urn/resolver.pl?urn=urn:nbn:de:0008-20090811196. – ISBN 9783940317483

Nickolai Zeldovich 2005
Nickolai Zeldovich, Ramesh C.: *Interactive performance measurement with VNCplay*. 2005. – URL http://people.csail.mit.edu/nickolai/papers/freenix2005-vncplay.pdf

Philipps 2010
PHILIPPS, Mario: *Entwurf und Implementierung eines Softwarearchivs für die digitale Langzeitarchivierung*. Online, http://hdl.handle.net/10760/14712. Juli 2010. – URL http://hdl.handle.net/10760/14712

Rechert u. a. 2009
RECHERT, Klaus ; SUCHODOLETZ, Dirk von ; WELTE, Randolph ; DOBBELSTEEN, Maurice van den ; ROBERTS, Bill ; HOEVEN, Jeffrey van der ; SCHRODER, Jasper: Novel Workflows for Abstract Handling of Complex Interaction Processes in Digital Preservation. In: *Proceedings of the Sixth International Conference on Preservation of Digital Objects (iPRES09)*, 2009

Ruzzoli 2010
RUZZOLI, Felix: *Ein Framework für die zustandsbasierte Fehlererkennung und -behandlung von interaktiven Arbeitsabläufen*. Online, http://www.ks.uni-freiburg.de/download/bachelorarbeit/WS09/03-VNC-FRuzzoli. März 2010. – URL http://www.ks.uni-freiburg.de/download/bachelorarbeit/WS09/03-VNC-FRuzzoli/Thesis-FRuzzoli-VNC.pdf

von Suchodoletz 2009
SUCHODOLETZ, Dirk von: *Funktionale Langzeitarchivierung digitaler Objekte – Erfolgsbedingungen für den Einsatz von Emulationsstrategien*. Cuvillier Verlag Göttingen, 2009

von Suchodoletz u. a. 2011
SUCHODOLETZ, Dirk von ; RECHERT, Klaus ; TCHAYEP, Achille N.: QEMU – A Crucial Building Block in Digital Preservation Strategies. In: MÜLLER, Wolfgang (Hrsg.) ; PÉTROT, Frederic (Hrsg.): *1st International QEMU Users' Forum – DATE 2011 Workshop*. Grenoble, France, 2011. – URL http://hdl.handle.net/10760/15406

Tchayep 2011
TCHAYEP, Achille N.: *Emulatoren-Testing für die digitale Langzeitarchivierung*. Online, http://www.ks.uni-freiburg.de/download/masterarbeit/SS11/. März 2011. – URL http://www.ks.uni-freiburg.de/download/masterarbeit/SS11/04-tchayep-qemu-testing/masterarbeit.pdf

Uhrig 2011
UHRIG, Volker: *View-Path Realizations for Obsolete Digital Objects*. Online, http://www.ks.uni-freiburg.de/download/bachelorarbeit/SS11/09-VUhrig-DOemuAccess/. September 2011. – URL http://www.ks.uni-freiburg.

de/download/bachelorarbeit/SS11/09-VUhrig-DOemuAccess/BA-VUhrig.pdf

Valizada 2011
VALIZADA, Isgandar: *Large-Scale, Transparent Format Migration System*. Online, http://eprints.rclis.org/handle/10760/16073. June 2011. – URL http://eprints.rclis.org/handle/10760/16073

Warnke und Ritzau 2010
WARNKE, Robert ; RITZAU, Thomas: *qemu-kvm & libvirt*. 4. Books on Demand GmbH, Norderstedt, 2010. – ISBN 9783837008760

Welte 2009
WELTE, Randolph: *Funktionale Langzeitarchivierung digitaler Objekte – Entwicklung eines Demonstrators zur Internet-Nutzung emulierter Ablaufumgebungen*. Südwestdeutscher Verlag für Hochschulschriften, 2009

Woods und Brown 2010
WOODS, Kam ; BROWN, Geoffrey: Assisted Emulation for Legacy Executables. In: *International Journal of Digital Curation* 5 (2010), Nr. 1. – URL http://www.ijdc.net/index.php/ijdc/article/view/153. – ISSN 1746-8256