JOHANNES KEPLER UNIVERSITÄT LINZ | JKU

TNF

Technisch-Naturwissenschaftliche Fakultät

Speichern von Webseiten zu forensischen Untersuchungszwecken

MASTERARBEIT

zur Erlangung des akademischen Grades

Master of Science

im Masterstudium

NETZWERKE UND SICHERHEIT

Eingereicht von:
Haudum Markus, 0555717

Angefertigt am:
Institut für Informationsverarbeitung und Mikroprozessortechnik (FIM)

Beurteilung:
o. Univ. Prof. Dr. Mühlbacher Jörg R.

Mitwirkung:
Assoz.-Prof. Mag. Dipl.-Ing. Dr. Michael Sonntag

Linz, Juni 2012

Kurzfassung

Möchte man den für die Außenwelt sichtbaren Stand einer Webseite abspeichern, so stehen hierfür, angefangen von der Benutzung von Archiven bis zum einfachen Ausdrucken von Bildschirmfotos, einige Möglichkeiten zur Verfügung. Sämtliche vorhandenen Systeme beziehungsweise Methoden leiden allerdings unter demselben Problem. Die Aktualität der Daten kann nicht garantiert werden aufgrund fehlender Caching-Informationen und des mangelnden Wissens über Proxys auf dem Weg vom Client zum Server. Bei Streitfällen, welche vor Gericht entschieden werden, ist es jedoch wichtig, dass zum Zeitpunkt der Sicherung der aktuelle Stand festgehalten wird und kein veralteter. Der Weg, auf dem diese Sicherung erstellt wurde, ist hierbei ebenso relevant, wichtig ist dabei vor allem die Unveränderlichkeit der Daten.

Diese Masterarbeit beschäftigt sich mit der Entwicklung einer Software, welche die Probleme existierender Technologien löst, und stellt ein Komplettpaket für die Speicherung von Webseiten zu forensischen Untersuchungszwecken zur Verfügung.

Hierfür wurde in Java die Anwendung Webstamp geschrieben, welche es den Benutzern ermöglicht, einzelne Webseiten abzuspeichern, und sich selbstständig darum kümmert, dass die angeforderten Inhalte aktuell sind und auch nicht zu einem späteren Zeitpunkt verändert werden können. Die forensischen Aspekte werden durch den Einsatz von Hashwerten und einem Zeitstempeldienst garantiert.

Abschließend folgt die Evaluierung der entwickelten Software anhand einiger ausgewählter Webseiten, welche ihre Stärken und Schwächen aufzeigen sollen. Daraus folgend werden im letzten Kapitel einige Verbesserungsvorschläge angeführt.

Abstract

When trying to store the visible state of a website there are a few possibilities available, starting from the usage of archives to the creation and printing of screenshots. All those methods and systems suffer from the same problem. They can't guarantee that the stored information is current because they don't provide any information about caching and it is unclear if the data came from proxies between the client and the server or from the original server. This information is very important in a law court, because it must be ensured that the collected data was current and not outdated. The way used to gather that information is also very important regarding the immutability of the collected data.

This master thesis describes the development of software to overcome the problems of existing technologies and provides a complete system for storing websites in a way that the collected data is of forensic value.

A java application called *Webstamp* was developed which allows the user to store specific websites. All the forensic aspects of immutable and most up to date data are taken over by the software automatically, which is ensured by hash-values and timestamp services.

At the end of this thesis an evaluation about the software is made based on a few selected websites. The collected information from this evaluation is then used to discuss improvements of the developed software.

Inhaltsverzeichnis

Abbildungsverzeichnis

Codeverzeichnis

Tabellenverzeichnis

1 Beweissicherung im Web

1.1 Einführung

Ziel dieser Masterarbeit ist die Entwicklung der Software Webstamp, die es ermöglicht, den aktuellen Zustand einer Webseite festhalten zu können. Wenn von einer Webseite gesprochen wird ist damit nicht der gesamte Domaininhalt gemeint, sondern eine einzelne Seite. Mithilfe der zu entwickelnden Software soll diese Seite abgespeichert werden können, um danach ihren Wert für die Beweissicherung zu erhalten.

Ein wesentlicher Aspekt der Beweissicherung ist hier die Aktualität der abgerufenen Informationen. Dies bedeutet, dass sichergestellt werden muss, dass der abgerufene Inhalt der Seite auch tatsächlich vorhanden ist und nicht etwa durch einen Proxy dargestellt wird, während der eigentliche Inhalt bereits vom entsprechenden Server entfernt oder verändert wurde.

Ein weiterer Punkt ist es, eine Möglichkeit zu geben, den gesicherten Inhalt in genau derselben Darstellung wiedergeben zu können, in der er vorgefunden wurde. Dies bedeutet, dass die gesicherte Webseite bei ihrem Aufruf möglichst genau dem Original entspricht. Dabei ist es wichtig zu garantieren, dass der gesicherte Inhalt nicht verändert wurde. Hierfür ist es erforderlich, eine elektronische Signatur über die gesicherten Daten zu erzeugen.

1.2 Aktuelle Situation

Aus den unterschiedlichsten Gründen ist es von Zeit zu Zeit erforderlich, den aktuellen Stand einer Webseite zu speichern. Beispielsweise könnte ein Web-Shop mit einem gefälschten Zertifikat auf der eigenen Webseite werben. Ein weiterer offensichtlicher Grund ist die Zurschaustellung von illegalen Inhalten, dabei kann es sich um Bilder, Videos oder Links handeln. Das zurzeit eingesetzte Verfahren, um solch offensichtliche Verstöße gegen das Gesetz zu dokumentieren, sieht aus wie folgt:

1. Die Webseite wird im Web-Browser aufgerufen.
2. Die Webseite wird auf einem lokalen Drucker ausgedruckt.
3. Eine Person, im Idealfall ein Anwalt, signiert den Ausdruck und versieht ihn mit dem aktuellen Datum.
4. Der Ausdruck muss jetzt händisch weitergereicht werden, bis er bei der dafür verantwortlichen Stelle ankommt.

1.3 Realisierung mit Webstamp

Der im vorangegangenen *Kapitel 1.2 Aktuelle Situation* beschriebene Vorgang ist weder schnell noch sicher, noch garantiert er ein korrektes Festhalten des aktuellen Zustandes der Webseite. Die Probleme beginnen bereits bei Punkt 1. - hier wird die Webseite mit einem nicht weiter definierten Browser aufgerufen. Abgesehen davon, dass die diversen Browser unterschiedlich auf manche Codestücke reagieren, kann ein Webserver relativ einfach so konfiguriert werden, dass bei verschiedenen Browsern - ja sogar bei unterschiedlichen Versionen der Browser - unterschiedliche Inhalte an den Client ausgeliefert werden. Ziel von Webstamp ist es, diesen Schritt über den Browser zu entfernen. Dies wird in der Form realisiert, dass nur die Antworten auf einfache HTTP-Requests gespeichert werden. So kann verhindert werden, Browser-spezifische Inhalte zu erhalten und zu speichern.

Punkt 2. betrifft Webstamp nicht, da das Ausdrucken des gesicherten Inhaltes einerseits nicht zielführend wäre und andererseits nicht Teil der Spezifikation ist. Das Ausdrucken wäre deswegen nicht zielführend, da das Ergebnis der Sicherung ein jederzeit verifizierbares Dokument sein soll. Druckt man die gesicherten Inhalte nun einfach aus, hat man dieselben Probleme wie vor dem Einsatz von Webstamp bestanden - ein nicht vertrauenswürdiges Dokument.

Punkt 3. steht und fällt mit der Vertrauenswürdigkeit, die mit einer bestimmten Person verknüpft ist. Wie kann man sicher sein, dass jene Person, die den Ausdruck erstellte, nicht doch etwas verändert hat? Vor dem Ausdrucken könnten Inhalte wie Bilder oder Text einfach entfernt oder verändert werden. Um diesen Unsicherheitsfaktor Mensch zu

entfernen, arbeitet Webstamp den gesamten Sicherungsprozess, inklusive dem Hinzufügen eines vertrauenswürdigen Zertifikates ohne die Zuhilfenahme des Benutzers automatisch ab.

Das Problem, das aus Punkt 4. hervorgeht, ist, dass garantiert werden muss, dass das zu übermittelnde Dokument vom Ausdruck bis zur endgültigen Verwendung nicht verändert werden kann. Da das Dokument hierbei durch die Hände von Menschen geht, ist diese Garantie nicht gegeben.

Eine weitere Schwierigkeit tritt auf, wenn das gesicherte Material an mehr als einem Ort gleichzeitig verwendet werden soll. Dies ist nicht möglich, da es sich, selbst wenn die betreffende Webseite mehrmals ausgedruckt wurde, jedes Mal um einen anderen Zeitstempel handelt. An allen Orten würden somit unterschiedliche Versionen der Sicherung verwendet werden.

Webstamp löst die Probleme aus Punkt 4., da die Sicherung elektronischer Natur ist und somit beliebig oft dupliziert werden kann, ohne Gefahr zu laufen, dass die Sicherung auf ihrem Weg verändert werden könnte. Dies wird durch die Validierungsfunktion von Webstamp garantiert, die zu jedem Zeitpunkt auf jedem Javafähigen Rechner mit Internetzugang ausgeführt werden kann.

2 Vorhandene Technologien

Dieses Kapitel gibt eine Übersicht über bereits vorhandene Technologien und beschreibt, weshalb sie die Anforderungen nicht erfüllen. Allgemeine Methoden wie das Ausdrucken von Webseiten und das Erzeugen von Screenshots, insbesondere deren Beweiskraft, werden ebenfalls behandelt.

2.1 Die Wayback-Maschine

Bei der Wayback-Maschine handelt es sich um ein von einer gemeinnützigen Organisation betriebenen Online-Portal, dessen Aufgabe darin besteht, das gesamte Internet zu archivieren. Archiviert werden hier abgesehen von Webseiten auch Texte, Audio-Dateien, Filme und Software. [9]

Abbildung 1 zeigt die Benutzeroberfläche dieser Anwendung, die im Wesentlichen aus einem Kalender besteht, der angibt, an welchen Tagen eine Sicherung durchgeführt wurde.

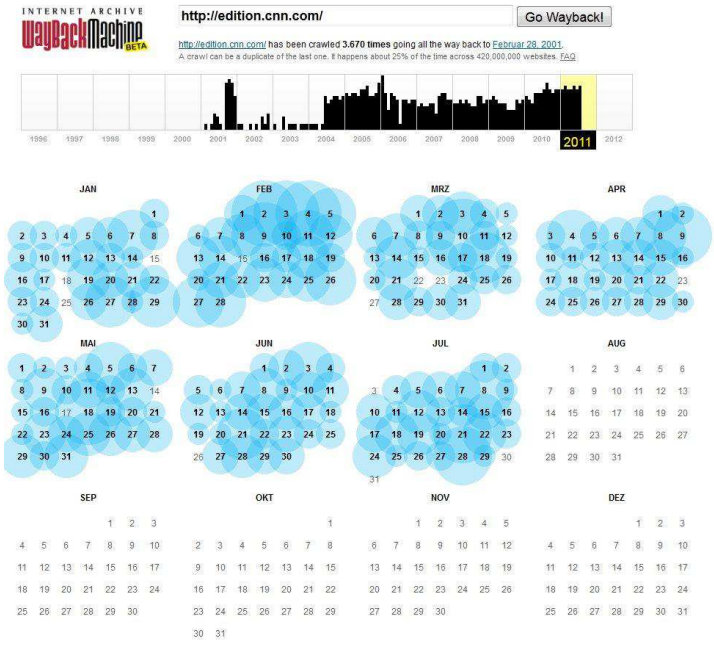

Abbildung 1: Wayback-Maschine [16]

Die Idee einer solchen Archivierung ist an sich sehr nützlich, falls man an der Vergangenheit einer Webseite interessiert ist. Allerdings sind hier einige sehr problematische Aspekte zu beachten, die deutlich machen, dass diese Plattform idealerweise nicht dafür herangezogen werden sollte, um Streitfälle vor Gericht zu entscheiden.

Jede gespeicherte Webseite wird von der Wayback-Maschine verändert, um sie auf dem eigenen Webserver anzeigen zu können. Dies hat den Grund, dass sich innerhalb eines Dokumentes beispielsweise Verweise auf die Original-Domain befinden, die umgeschrieben werden müssen, um den Datenbestand zum Zeitpunkt der Sicherung anzeigen zu können. Ansonsten könnte beim Aufruf der Webseite der aktuelle an Stelle des archivierten Datenbestandes angezeigt werden. „Mittels eines speziellen Parameters (anhängen von „id_“ an die Dokumentennummer) ist jedoch ein Zugriff auf die ‚Originalversion' möglich. Hier ist zu beachten, dass dann etwaige externe Elemente, z. B. Bilder, von der aktuellen Webseite stammen, da die Links nicht mehr auf das Archiv zeigen.“ [9]

Der Inhalt, der sich unter einer Domain befindet, sprich die gesamte Webseite, wird nicht zum selben Zeitpunkt gespeichert. Dies hat zur Folge, dass sich das Datum der einzelnen Seiten voneinander unterscheiden kann. Zusätzlich können die Elemente, die innerhalb einer Seite dargestellt werden, ebenfalls von unterschiedlichen Zeitpunkten stammen. Das kann beispielsweise bei Grafiken der Fall sein oder bei Frames, wo das angegebene Datum nicht dem Inhalt des Frames entspricht, sondern dem Frame an sich. [9]

Skripte, speziell jene, welche Inhalte dynamisch nachladen, sind immer problematisch, da sie entweder nur den aktuellen Datenbestand darstellen oder gar keinen, abhängig davon, ob die entsprechenden Skripte noch existieren oder nicht.

Es sollte hier noch angemerkt werden, dass die optische Präsentation der archivierten Version der Webseite http://edition.cnn.com ident mit der durch Webstamp durchgeführten Sicherung ist. Die problematischen Bereiche dieser Webseite werden im *Kapitel 9.4.3 CNN* behandelt.

„Zusammengefasst ist festzustellen, dass eine archivierte Seite zwar als Beweismittel potenziell geeignet ist, aber eine genaue Untersuchung des Quelltextes (→ JavaScript) sowie aller eingebetteten Elemente (Bilder, Subframes, dynamischer Inhalt), sofern relevant, durch Experten erforderlich ist.“ [9]

Aus den genannten Gründen kann die Wayback-Maschine nicht die Aufgaben von Webstamp übernehmen, allerdings konnten einige der Ansätze, wie das Umschreiben von Links, übernommen werden.

2.2 Der Google-Cache

Damit Google seinen Benutzern die Möglichkeit zum Durchsuchen des Internets geben kann, müssen die entsprechenden Webseiten erst indexiert werden. Bei den eigentlichen Suchanfragen stellt Google die Möglichkeit zur Verfügung, eine alte Version einer Webseite betrachten zu können. Diese alten Versionen werden als der Google-Cache bezeichnet. Nun stellt sich die Frage, ob dieser Cache im Rahmen einer Beweisführung Verwendung finden könnte.

Problematisch ist, dass Google nur eine Kopie des Textes, sprich des HTML-Codes, anlegt. Weiter wird die Webseite durch das Hinzufügen von eigenem Code, einem Header, verändert. Abgesehen von diesem Header, der sich nicht auf das restliche Dokument auswirkt, werden allerdings keine Veränderungen vorgenommen. „Daraus folgt jedoch im Umkehrschluss, dass darauf enthaltene Links zur aktuellen Seite führen und alle eingebetteten Elemente immer die .aktuellen‘ sind: Es wird lediglich der Webseiten-Quelltext archiviert, nicht jedoch Links, externe Stylesheets/JavaScript, Applets etc.“ [9]

Das Fehlen der eingebetteten Inhalte einer Webseite ist der Hauptgrund, warum der Google-Cache nicht zur Beweisführung herangezogen werden kann. Es wäre beispielsweise unmöglich zu beweisen, dass eine im Cache angezeigte Grafik ident mit jener zum Zeitpunkt der Sicherung ist. Ein reales Anwendungsbeispiel hierfür wäre beispielsweise die Einbindung einer Grafik, welche das aktuelle Wetter anzeigt. Im Google-Cache würde nun

nicht die Grafik zum Zeitpunkt der Sicherung angezeigt werden, sondern jene, die sich aktuell auf dem Server befindet bzw. auf einem Proxy.

2.3 Das österreichische Webarchiv

Der Gedanke, welcher hinter dem österreichischen Webarchiv steht, ist die Sammlung und Archivierung des gesamten nationalen Webspace zur späteren Betrachtung. [7] Dies betrifft alle „.at-Domains“ sowie ausgewählte Webseiten mit Bezug zu Österreich. Damit soll eine Möglichkeit geschaffen werden, alte Versionen von Webseiten sowie nicht mehr vorhandene betrachten zu können. Falls keine besonderen Gründe vorliegen, so wird ein Update einer Webseite allerdings nur alle zwei Jahre durchgeführt, was im Weiteren zu einem äußerst magerem Datenbestand innerhalb einer Webseite führt. Ein Feature, das diese Software von den anderen abhebt, ist, dass die robots.txt ignoriert wird, die normalerweise von Seitenbetreibern genutzt wird, um öffentliche Inhalte für Suchmaschinen zu sperren.

Im Gegensatz zum Google-Cache und zur Wayback-Maschine ist es nicht möglich, dieses Archiv online einzusehen. Die einzige Möglichkeit, an den Datenbestand zu kommen, ist die Verwendung spezieller Terminals, die jeden Export, abgesehen vom Ausdrucken, unterbinden.

Das größte Problem dieses Archivs ist, dass domainfremde Inhalte nicht abgespeichert werden. Das bedeutet, dass beispielweise ausgelagerte Bilddateien oder Werbung sich nicht in der Sicherung wiederfinden. Webstamp löst dieses Problem, indem dem Benutzer die Option gegeben wird, externe Inhalte zu ignorieren oder zu speichern.

Die Benutzung des Archivs ist über ein Suchformular gelöst, dass im Gegensatz zu einer Suchmaschine nicht Dokumente liefert, die mit dem Suchbegriff übereinstimmen, sondern Domains, in denen der Suchbegriff vorkommt.

2.4 Screenshots

Beim Gedanken daran, den aktuellen Zustand einer Webseite festzuhalten, liegt es nahe, es mit einem Screenshot zu versuchen. Das Hauptproblem hierbei ist die relativ triviale Möglichkeit, den Inhalt vor und nach dem Anfertigen des Screenshots verändern zu können. Dafür muss nicht einmal sehr viel technisches Wissen vorhanden sein, ein einfaches Browser-Plugin reicht bereits aus, um Veränderungen an der aktuell dargestellten Version vornehmen zu können. Somit wäre es ein Leichtes, einzelne Textstellen oder Grafiken auszutauschen.

Ein ebenfalls nicht zu unterschätzendes Problem ist, dass diese Screenshots nur eine an den in Verwendung befindlichen Browser angepasste Version anzeigen. Wünschenswert wäre jedoch eine vom Browser unabhängige Sicherung beziehungsweise die Möglichkeit, den gewünschten Browser zumindest auswählen zu können.

2.5 Ausdruck

Ebenso wie ein Screenshot, so hat auch ein Ausdruck wenig Beweiskraft vor Gericht, da dieser zuvor relativ einfach manipuliert werden kann. „Insbesondere ist das Ausdrucksdatum nicht ersichtlich. Es wird zwar meist eine Datums- und Zeitangabe mit ausgedruckt, diese beruht jedoch lediglich auf der lokal eingestellten Zeit (=trivial veränderbar).“[9]

Im Folgenden werden einige Methoden aufgelistet, welche es einer Person ermöglichen, den Ausdruck einer Webseite an die eigenen Wünsche und Vorstellungen anzupassen.

- Durch die Verwendung eines Plugins kann die Webseite bereits im Browser verändert werden.
- Wurde die Webseite abgespeichert, so können der HTML-Code sowie etwaige Style-Informationen leicht angepasst werden. Hierfür müssen nur die entsprechenden Dokumente in einem üblichen Text-Editor angepasst werden.

- Beim Ausdruck eines Screenshots einer Webseite können Bildbearbeitungsprogramme herangezogen werden, um den Screenshot vor dem Ausdruck zu verändern.

2.6 Speicherung durch Browser

Das Abspeichern von Webseiten zählt mittlerweile zur Standardfunktionalität von Web-Browsern. Dem Benutzer wird hierbei meist die Wahl gelassen, ob er nur den vorhandenen Text, sprich den HTML-Code, sichern möchte, oder ob er auch Kopien von Ressourcen wie Grafiken und Skripten anlegen will.

Im Folgenden werden einige Probleme aufgelistet, die durch die Speicherung im Browser entstehen können.

- Inhalte, die nicht von derselben Domain stammen wie die zu speichernde URL, werden meist in einem einzigen Ordner für externe Dateien zusammengefasst. Dies macht es unmöglich, ohne zusätzliche Informationen die Herkunft dieser Dateien bestimmen zu können.
- Die Zeitstempelinformationen, die mit der Sicherung erzeugt wurden, können relativ einfach manipuliert werden. Das Kopieren des entsprechenden Ordners würde hierfür bereits ausreichen.
- Es gibt keine Garantie dafür, dass es sich bei den gesicherten Daten auch um die aktuellen Daten handelt. Es könnte ohne Weiteres sein, dass die gesicherten Inhalte von einem Proxy stammen und nicht vom Original-Server.

2.7 Zusammenfassung

Die Systeme und Methoden, die in diesem Kapitel vorgestellt wurden, haben im Grunde eines gemeinsam. Ihr Zweck ist es, eine im Internet verfügbare Webseite so abzuspeichern, dass sie später erneut betrachtet werden kann. Der Schwerpunkt liegt in einer möglichst genauen Darstellung der Kopie und nicht in der Nachvollziehbarkeit der Sicherung. Es wird

ein fertiges Produkt (die gesicherte Webseite) ausgeliefert, ohne irgendeine vertrauenswürdige Möglichkeit zu geben, um feststellen zu können, wie es zu dieser Kopie kam.

Dies ist die wesentliche Stelle, an der sich die im Rahmen dieser Masterarbeit entwickelte Software (Webstamp) von den vorhandenen Systemen unterscheidet. Mithilfe dieser Software ist es nicht nur möglich, eine aktuelle Version einer Webseite und ihrer Elemente zu erhalten; auch der Weg, der zur Sicherung führte, ist dadurch nachvollziehbar.

3 Inhalt einer Webseite

Dieses Kapitel beschäftigt sich mit den Bestandteilen von Webseiten und damit, wie diese möglichst unverändert festgehalten und zu einem späteren Zeitpunkt wiederhergestellt werden können.

Eine Webseite an sich ist ein dynamisches Objekt, dynamisch im Hinblick darauf, dass sich ihr Aussehen und ihre Funktionalität durch die Interaktion mit dem Benutzer verändern können. Für dieses Projekt fällt der dynamische Aspekt weg, da es die Aufgabe von Webstamp ist, eine statische Repräsentation der Webseite zu sichern, die durch eine eindeutige URL repräsentiert wird. Das bedeutet, dass einzelne Seiten abgespeichert werden, nicht aber der Inhalt einer gesamten Webseite.

Eine Seite ist ein Dokument, bestehend aus einer HTML-Datei und allen damit in Verbindung stehenden Dateien für Skripte und Grafiken. [13]

3.1 Bestandteile einer Webseite

Sobald man damit beginnt, sich Gedanken darüber zu machen wie der eigentliche Sicherungsvorgang ablaufen soll, muss man auch unweigerlich einige Überlegungen darüber anstellen, aus welchen Bestandteilen eine Webseite besteht. Im Weiteren führt dies in diesem Kapitel zu einer Einteilung der einzelnen Bestandteile, die im Hinblick auf die letztendliche Verwendung hin, das Speichern, optimiert ist.

Eine Webseite wird demzufolge von mir in drei grundlegende Bestandteile unterteilt:

- HTML-Quellcode
- Visuelle Repräsentation anhand von Stylesheets oder Style-Informationen
- Ressourcen.

3.1.1 HTML-Quellcode

Der eigentliche Source-Code, der eine Webseite ausmacht, ist nicht verfügbar, da sich dieser auf einem nicht zugänglichen Server befindet. Der Quellcode kann in Java, PHP, ASP oder einer anderen Script- oder Programmiersprache verfasst worden sein. Der Informationsgehalt, der aus diesen sich auf dem Server befindlichen Dateien gewonnen werden könnte, ist allerdings gar nicht erforderlich, da lediglich das Endprodukt, die ausgelieferte Seite, von Interesse ist. Zum Abspeichern ist somit lediglich der HTML-Code nötig, der mithilfe der Skript-Dateien am Server generiert und an den Client nach einem HTTP-Request ausgeliefert wird.

Das Erstellen einer Sicherung des HTML-Quellcodes ist einfach zu realisieren, da diese Funktion beim Aufrufen einer Webseite von jedem Browser durchgeführt wird und somit mit Leichtigkeit implementiert werden kann.

3.1.2 Stylesheet

Im Bereich des Webdesigns wird dazu geraten, Style-Informationen nicht in den HTML-Quellcode einzubetten, sondern spezielle Dokumente, die Stylesheets genannt werden, dafür zu verwenden. Stylesheets sind im Grunde Ressourcen, auf die im Header eines HTML-Quellcodes verwiesen wird. Dieser Verweis kann allerdings auch an anderer Stelle auftreten, beispielsweise in der Mitte des Codes durch eine *include()* Anweisung. Durch die Einbindung eines solchen Stylesheets ist es dem Webdesigner möglich, das Verhalten und Aussehen von HTML-Tags zu verändern. Beispielsweise könnte über jedem Element mit einer bestimmten ID ein Informationsfenster angezeigt werden. Codestück 1 zeigt, wie mehrere Stylesheets im Bereich eines HTML-Headers eingebunden werden können.

```
<style type="text/css" media="all">
        @import url("http://hakkon-aetterni.at/modules/system/system.base.css?lnxhcd";
        @import url("http://hakkon-aetterni.at/modules/system/system.menus.css?lnxhcd";
        @import url("http://hakkon-aetterni.at/modules/system/system.mess.css?lnxhcd";
        @import url("http://hakkon-aetterni.at/modules/system/system.theme.css?lnxhcd";
</style>
```

Codestück 1: Stylesheet, Import-Anweisung

Zur Sicherung der visuellen Repräsentation ist es nötig, das entsprechende Stylesheet, falls eines vorhanden ist, ebenfalls zu sichern. Dabei gilt es zu beachten, dass eine Webseite eine Vielzahl an Stylesheets referenzieren kann. Es ist beispielsweise üblich ein spezielles Stylesheet zum Ausdrucken des entsprechenden Dokumentes zur Verfügung zu stellen. Natürlich können Style-Informationen auch innerhalb des HTML-Quellcodes beziehungsweise innerhalb von Java-Skripten vorkommen. In diesen Fällen kommen speziell für diesen Zweck konstruierte reguläre Ausdrücke zum Einsatz, die darauf abzielen, nach Möglichkeit alle absoluten und relativen Links aus einem unstrukturierten Dokument zu extrahieren.

3.1.3 Ressourcen

Ressourcen sind alle Bestandteile einer Webseite, auf die im HTML-Quellcode verwiesen wird, mit der Einschränkung, dass deren Inhalt nicht Teil des Quellcodes selbst ist. Dies betrifft alle Arten von Grafiken, Videos und Skripten.

Relativ einfach lassen sich jene Ressourcen abspeichern, die im Quellcode durch einen relativen oder absoluten Pfad angegeben werden. Problematisch wird es, falls diese Pfade nicht statisch sind sondern dynamisch, oder z. B. durch die Zuhilfenahme von JavaScript zusammengesetzt werden.

Ohne eine größere zusätzliche Logik ist es nicht möglich, Ressourcen zu speichern, deren Pfad nicht vollständig ist, egal ob relativ oder absolut.

Das Problem von nicht speicherbaren Ressourcen tritt zum Beispiel auf der Webseite von CNN vom 25.10.11 auf. Codestück 2 zeigt den entsprechenden Quellcode.

```
<script type="text/javascript">
	var cnnCdnPath =
	'http://i.cdn.turner.com/cnn/.element/img/3.0/global/header/intl/';

	var cnnIntlBanners = [
		'img src="' + cnnCdnPath + 'hdr-globe-west.gif">',
		'img src="' + cnnCdnPath + 'hdr-globe-central.gif">',
		'img src="' + cnnCdnPath + 'hdr-globe-east.gif">',
	];

	var cnnRNum = Math.floor(Math.random() * cnnIntlBanners.length);
	document.getElementyById('hdr-banner-title').innerHTML = cnnIntlBanners[cnnRNum];

</script>
```

Codestück 2: Dynamische Ressourcen-Pfade anhand von CNN vom 25.10.11

Dieses Skript, das Bestandteil der CNN-Startseite ist, ist dafür verantwortlich, ein Bild der Weltkugel darzustellen, mit dem Fokus auf Amerika, Europa oder Asien. Die Pfade zu den einzelnen Grafiken werden hierbei dynamisch generiert. Mittels eines Zufallsalgorithmus wird danach eine der drei Grafiken ausgewählt.

Dynamische Pfade, die vom Browser zusammengesetzt werden, wie es in Codestück 2 der Fall ist, können von Webstamp nicht gesichert werden.

3.2 Welche Daten werden gesichert

Nach dem in den vorangegangenen Unterkapiteln die Strukturierung und Erklärung der zu sichernden Elemente stattfand, wird nun festgehalten, welche Elemente gespeichert werden müssen, um eine forensisch verwertbare Kopie des Originaldokuments zu erhalten.

- HTML-Quellcode
    - Der HTML Quellcode wird zur Strukturierung des Inhalts der Webseite herangezogen und muss somit gespeichert werden. Durch ihn wird das Grundgerüst einer Webseite definiert.
- Stylesheet
    - Stylesheets legen fest, wie die einzelnen Elemente auf der Webseite aussehen. Es folgt ein Auszug der Möglichkeiten:

        - Festlegen der Größe und Art der verwendeten Schrift.
        - Bildoperationen wie das Verändern der Größe und das Hinzufügen von Umrahmungen.
        - Ein- und Ausblenden von einzelnen Elementen.
    - Um die Aufmachung des Originaldokuments zu erhalten, müssen alle Style-Informationen gesichert werden.
- Ressourcen
    - Als Ressourcen werden hier jene Inhalte bezeichnet, die weder zur Strukturierung noch zum Anpassen des Aussehens herangezogen werden, abgesehen von JavaScript, das beide Aufgaben übernehmen könnte. Im Grunde handelt es sich hierbei um Multimedia-Inhalte sowie JavaScript. Mögliche Ressourcen sind Bilder (JPEG, BMP, PNG, GIF), Videos (Flash), Musik (WMV, MP3, …) und JavaScripte.

3.2.1 Wie wird gesichert

Zu Beginn gibt der Benutzer eine URL ein und legt zwei verschiedene Optionen fest.

- Option 1: Sollen Ressourcen von fremden Domains gesichert werden?
- Option 2: Sollen HTML-Inhalte aus Skripten gesichert werden?

Option 1 behandelt eine Situation, wie sie vor allem auf größeren Webseiten immer häufiger vorkommt. Hierbei geht es darum, dass Ressourcen aus Performanz-Gründen oder auch aus rechtlichen Gründen auf verschiedene Server ausgelagert werden. Beispielsweise könnte die Domain http://www.example.at alle ihre Bilder auf einem Server mit der Domain http://www.images.example.at ablegen. Falls es der Benutzer wünscht, so kann er zu den Inhalten der angegebenen Domain auch domainfremde Inhalte speichern.

Option 2 soll verhindern, dass der Sicherungsvorgang zu einem Webcrawling-Vorgang entartet. Möchte der Benutzer zum Beispiel das Ergebnis einer Google-Suche sichern, so

besteht kein Interesse daran, auch die Inhalte der Suchergebnisse zu speichern. Google hingegen führt die Links zu all den gefundenen Webseiten aus Gründen der Benutzerfreundlichkeit (dynamisches Menü) in einem JavaScript auf. Wenn es nun zum Durchsuchen dieses Skriptes käme, so würden die Suchergebnisse als neue Inhalte erkannt und ebenfalls gesichert werden. Aus diesem Grund wird dem Benutzer die Option gegeben, ob er die URLs die auf eine HTML-Seite verweisen aus JavaScript-Dateien extrahieren möchte oder auch nicht.

Nach der Angabe der URL und dem Festlegen der beiden Optionen wird die HTML-Datei, die sich unter der URL befindet, mittels eines HTTP-Requests gespeichert. Anschließend wird die gesicherte Datei nach URLs durchsucht. Sämtliche gefundenen Inhalte werden, entsprechend den beiden zuvor gesetzten Optionen, gesichert. Handelt es sich bei den gesicherten Daten um HTML-, CSS-, oder JavaScript-Dateien, so werden diese ebenfalls erneut nach URLSs durchsucht.

Abschließend werden alle gespeicherten Daten unter Zuhilfenahme eines Zeitstempeldienstes elektronisch signiert.

4 Proxy

4.1 Einführung

Dieses Kapitel soll einen Überblick darüber geben welche Vor- und Nachteile durch den Einsatz von Proxys entstehen, was ein HTTP-Proxy ist und wieso es wichtig ist auf diese bei der Entwicklung von Webstamp einzugehen.

Grundsätzlich werden hier nur Proxys behandelt, die Caching-Mechanismen einsetzen, da einfache Proxys zur Weiterleitung keinen Einfluss auf die Funktionen von Webstamp haben.

4.2 Kategorisierung

Proxy-Server können je nach Konfiguration in eine von drei verschiedenen Kategorien eingeteilt werden. Hierbei wird zwischen L1-, L2- und L3-Proxys unterschieden.

- L1 (Elite-Proxy)
    - Hierbei handelt es sich um einen Proxy zur Anonymisierung der einzelnen Benutzer durch das Verstecken ihrer IP-Adressen. L1 Proxys geben sich aus Sicht des Servers als End-Client aus.
- L2 (Anonyme Proxy)
    - Ähnlich wie L1-Proxys verstecken auch L2-Proxys die IP-Adresse des Clients. Allerdings teilen solche Proxys den Servern mit, dass es sich bei dem Anfragenden um einen Proxy handelt und nicht um einen End-Client.
- L3 (Transparente Proxy)
    - Diese Proxys bieten keine Form der Anonymisierung, da sie dem Server die IP-Adresse des End-Clients übermitteln und klarstellen, dass sie die Aufgabe eines Proxys übernehmen. Diese Art von Proxy wird meist mit L1- und L2-Proxys kombiniert. [1]

4.3 HTTP-Proxy

Für den Rahmen dieser Arbeit beschränke ich mich hier auf reine HTTP-Proxys beziehungsweise Caching-Proxys. Diese Proxys können als L1-, L2- und als L3-Proxy auftreten und führen zu einem großen Problem, wenn es darum geht, den aktuellen Zustand von Daten auf einem Webserver festzuhalten. Im folgenden *Unterkapitel 4.4 Vor- und Nachteile* wird darauf genauer eingegangen.

Abbildung 2 zeigt die Position und Aufgabe eines Caching- bzw. HTTP-Proxys. Hierbei soll veranschaulicht werden, dass eine direkte Kommunikation zwischen Client und Server im Internet kaum vorkommt und sich zwischen diesen beiden Stationen einer oder mehrere Proxys befinden können.

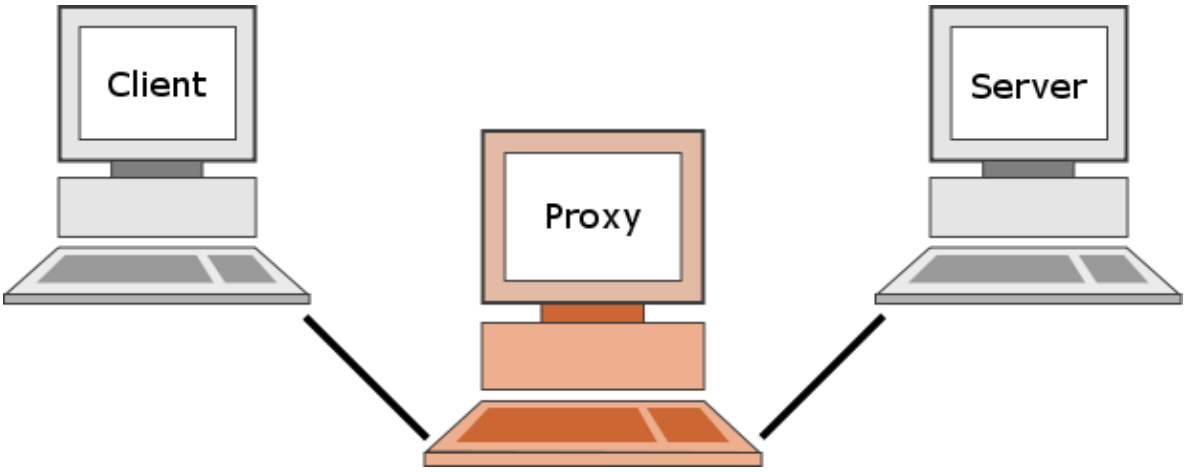

Abbildung 2: Konzept eines HTTP-Proxys

Ein Proxy ist im Grunde eine Zwischenstation, die auf dem Weg vom Client zum Server und auch wieder zurück besucht wird. Hierbei analysiert der Proxy die Anfrage des Clients und überprüft, ob der angeforderte Inhalt bereits im eigenen System vorhanden ist. Ist dies der Fall kann, der Proxy den zwischengespeicherten Inhalt direkt an den Client ausliefern oder, falls er der Meinung, ist dass die zwischengespeicherten Inhalte nicht mehr aktuell sind, die geforderten Daten direkt vom Server holen.

Wie Webstamp mit Proxys und im Speziellen mit diesen Caching-Mechanismen umgeht, wird in *Kapitel 5 Caching* beschrieben.

4.4 Vor- und Nachteile

Die drei wesentlichen Vorteile, die durch die Verwendung eines Caching-Proxys entstehen sind:

- Die Verzögerung beim Zugriff auf eine Ressource wird reduziert.
- Die allgemeine Last im gesamten Netzwerk sowie die Last am Ziel-Server werden reduziert.
- Bei Ausfall des Ziel-Servers können die gewünschten Inhalte immer noch abgerufen werden.

Die potenziellen Nachteile, die sich durch den Einsatz eines solchen Caching-Mechanismus ergeben, sind:

- Es werden veraltete Versionen von Dokumenten ausgeliefert, falls sich das Dokument am Ziel-Server ändert und der Proxy diese Information noch nicht erhalten hat.
- Die Verzögerung für die Anfrage von nicht gecachten Inhalten dauert länger als die Anfrage für gecachte Inhalte.
- Der administrative Aufwand ist größer, und zusätzlich kostet dieser Einsatz einen gewissen Festplattenspeicher beziehungsweise einen zusätzlichen Computer. [2]

4.5 Proxys und Webstamp

“Since the majority of Web documents requested are static documents (i. e. home pages, audio and video files), caching at various network points provides a natural way to reduce web traffic. A common form of web caching is caching at HTTP Proxys, which are intermediaries between browser processes and web servers on the Internet […]” [3]

Das Themengebiet Proxy hat für diese Arbeit einen sehr hohen Stellenwert, da die Korrektheit des Endergebnisses, eine vertrauenswürdige und aktuelle Sicherung, davon

abhängt, ob es gelingt, die auftretenden Proxys dazu zu bringen, nur aktuelle und keine zwischengespeicherten Daten auszuliefern.

Eines der grundlegenden Features von Webstamp ist es, sicherstellen zu können, dass als Antwort auf einen HTTP-Request immer das aktuellste Dokument vom gewünschten Server erhalten wird und nicht etwa eine zwischengespeicherte und möglicherweise veraltete Version. Ohne diese Funktionalität wäre die fertige Sicherungskopie nicht aussagekräftig, da der Server-Betreiber damit argumentieren könnte, dass die gespeicherten Inhalte sich gar nicht mehr auf dem Original-Server befunden haben und es sich bei der gesicherten Version um eine alte Version aus dem Cache eines Proxys handelt.

5 Caching

5.1 Einführung

Mit der rasanten Entwicklung des Internets war es nötig, neue Methoden zu entwickeln und alte Methoden an die neue Technologie anzupassen. Um die Performanz zwischen dem Content-Provider und dem Client zu erhöhen, wurde damit begonnen Caching Verfahren einzusetzen.

“Expectations of scalability and performance have made caching and replication common features of the infrastructure of the Web. By directing the workload away from possibly overloaded origin Web servers, Web caching and replication address Web performance and scalability from the client side and the server side, respectively. Caching stores a copy of data close to the data consumer (e.g., in a Web browser) to allow faster data access than if the content had to be retrieved from the origin server. Replication, on the other hand, creates and maintains distributed copies of content under the control of content providers.” [4]

Wie aus dem Zitat von Rabinovich hervorgeht, wird das Zwischenspeichern von Inhalten in zwei verschiedene Bereiche unterteilt. In die Speicherung beim Client und die Speicherung auf Zwischenstationen, sogenannten Proxys. Der für diese Arbeit relevante Aspekt ist die Speicherung auf Proxys. Die clientseitige Speicherung ist nicht relevant, da Webstamp zur Ausführung keinen Webbrowser benötigt und selbst keine Caching-Methoden einsetzt. Im Folgenden ist, falls von Caching gesprochen wird, immer das Zwischenspeichern auf Proxys gemeint.

5.2 Szenario

Folgendes Szenario soll die Vorgänge beschreiben, die auftreten, wenn zwei Benutzer A und B nacheinander denselben Inhalt von einem Server anfordern und sich auf dem Weg beider Nutzer ein Proxy befindet. Abbildung 3 zeigt dieses Szenario. Es gilt zu beachten

dass die Punkte *Check: Ressource* die Momente am Proxy darstellen, zu denen er versucht, den geforderten Inhalt von A beziehungsweise B im Cache zu finden.

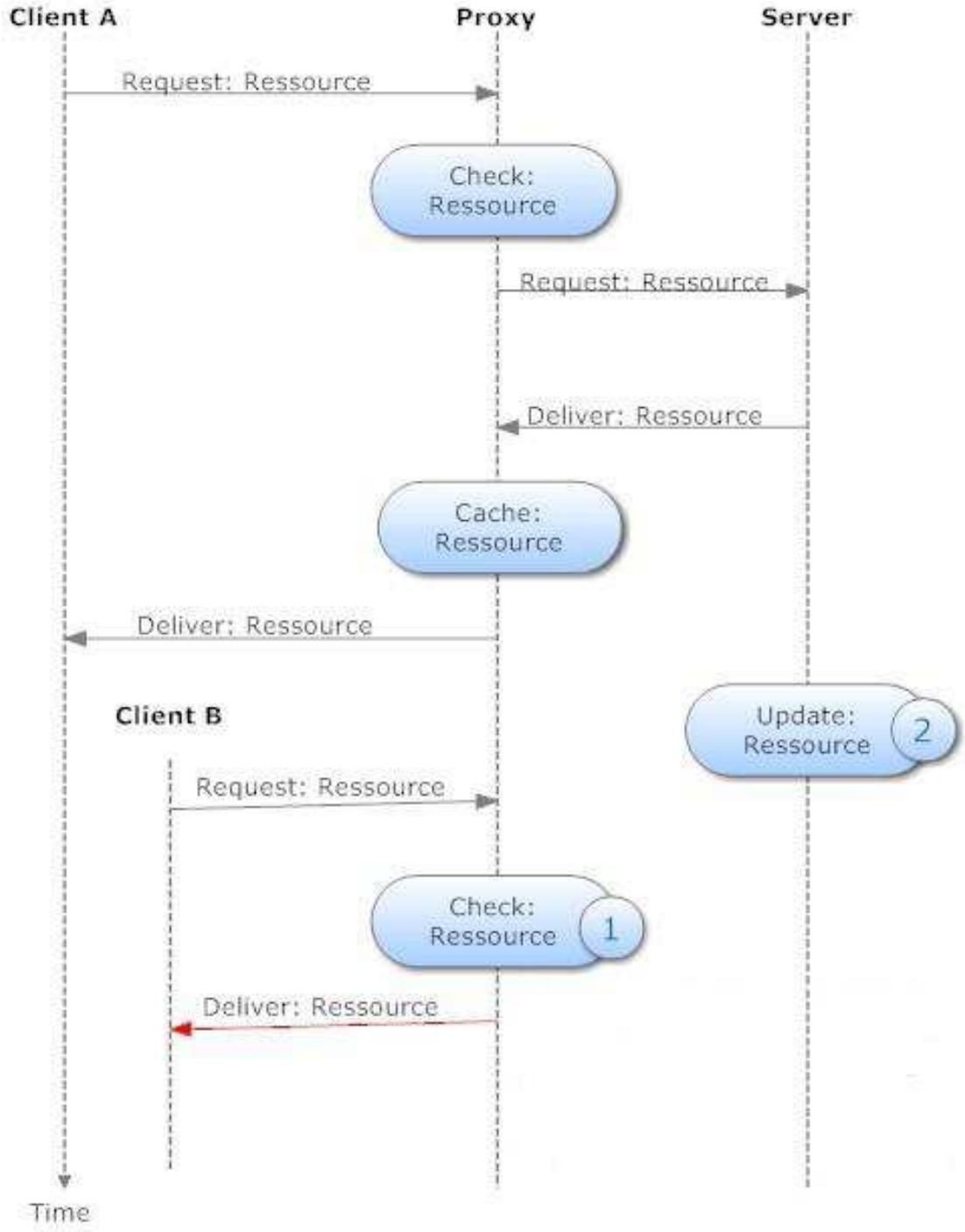

Abbildung 3: UML, Caching-Problem

Stellt Client A eine Anfrage an den Server, so wird diese nicht direkt an den Server gesendet, sondern über eine Zwischenstation, einen sogenannten Proxy, weitergeleitet. Der Server selbst schickt die angeforderte Information zurück an den Proxy, und dieser leitet sie an Client A weiter. In diesem Fall bekommt Client A die aktuellste Ressource geliefert.

Falls ein weiterer Client B nun nach kurzer Zeit dieselbe Anfrage wie Client A an den Server schickt, so stellt der Proxy fest, dass er die Informationen bereits hat, da er die geforderte Ressource bereits an Client A ausgeliefert und gespeichert hat. Somit ruft der Proxy nicht die Information vom Server ab sondern schickt die gecachte Version der angeforderten Ressource einfach direkt an Client B. Dadurch wird der Server entlastet. In Abbildung 3 wurde der Moment, in dem der Proxy feststellt, dass sich bereits eine Kopie der angeforderten Ressource im Cache befindet, mit der Nummer 1 markiert

Das Problem, das sich hieraus offensichtlich ergibt, ist, dass der Server die angeforderte Ressource inzwischen verändert haben könnte und Client B somit eine veraltete Version erhält. In Abbildung 3 wird der Moment, in dem der Server die entsprechende Ressource auf den aktuellsten Stand bringt, mit der Nummer 2 markiert.

5.3 Caching umgehen

Im Folgenden werden zwei Möglichkeiten beschrieben, um das Caching zu umgehen, mit dem Ziel, bei jeder Anfrage die aktuellste Ressource zu erhalten. Für Codebeispiele wird die Programmiersprache Java verwendet, dies ist zulässig, da Caching unabhängig von der verwendeten Technologie, mit der die Ressourcen angefordert werden, funktioniert.

5.3.1 Option 1: Cache-Direktiven

Option 1 basiert darauf, bei jeder Anfrage die Bitte zu äußern, keine gecachte Version zu erhalten. Wohlgemerkt handelt es sich hier um eine Bitte, welche die Proxys ablehnen können. Ein standardmäßig konfigurierter Proxy wird sich aber in den meisten Fällen daran halten, allerdings darf man sich darauf nicht verlassen. Das folgende Codestück 3 zeigt eine Implementierung dieses Szenarios.

```
URL u = new URL("http://www.google.at");
HttpURLConnection huc =  (HttpURLConnection)  u.openConnection();

huc.setRequestMethod("GET");
huc.setRequestProperty("User-Agent", userAgent);
huc.addRequestProperty("Cache-Control", "no-cache, no-store, must-revalidate");
huc.setUseCaches(false);


huc.connect();
```

Codestück 3: Caching-Direktiven

Die Klasse URL repräsentiert eine Ressource und die Klasse *URLConnection* die Verbindung zu derselben. Mittels der Funktion setUseCaches(false) wird die Bitte geäußert, die Ressource direkt vom Server anzufordern. Dies bewirkt, dass im HTTP-Header der Wert Cache-Control auf no-cache gesetzt wird. Anschließend kann der eigentliche Download durchgeführt werden.

5.3.1.1 Option 1: Die Problematik

Das grundlegende Problem dieser Methode wird im RFC 3143, *Kapitel 3.2.1 Wie wird gesichert* erörtert. Es folgt ein Auszug aus diesem RFC, der dieses Problem beschreibt.

“Clients may receive data that is not synchronized with the origin even when they request an end to end refresh, because of the lack of inclusion of either a "Cache-control: no-cache" or "must-revalidate" header. These headers have no impact on origin server behavior so may not be included by the browser if it believes it is connected to that resource. Other related data implications are possible as well.” [5]

HTTP wurde so entwickelt, dass der Benutzer immer weiß, ob er direkt mit dem Server kommuniziert oder mit einem Proxy. Die sogenannten Interception-Proxys machen diesen Grundgedanken allerdings zunichte. Hierbei handelt es sich um für den Client unsichtbare Proxys. Glaubt der Client nun, er befindet sich im direkten Kontakt mit dem Server, obwohl er es mit einem Interception-Proxy zu tun hat, so verzichtet er möglicherweise auf das Senden wichtiger Cache-Direktiven. Dies hat zur Folge, dass der Client eine veraltete Version erhält.

Im zuvor erwähnten Java-Beispiel wird zwar immer darum gebeten, eine aktuelle Version zu erhalten, allerdings wird dies nicht garantiert, da jeder Proxy die potenzielle Möglichkeit hat, diese Cache Direktiven einfach zu ignorieren und so den HTTP-Header zu verändern.

5.3.2 Option 2: Parameter anhängen

Wie in der Einführung bereits erwähnt, liefert ein Proxy nur dann eine gecachte Version einer Ressource, wenn diese bereits von jemand anderem angefordert wurde. Ein wesentlicher Punkt, um diese Option verstehen zu können, ist, dass die URLs in Tabelle 1 dieselbe Ressource repräsentieren, sie vom Proxy aber als unterschiedlich erachtet werden.

URLs ohne Parameter	
Ursprüngliche URL	www.google.at
Angepasste URL	www.google.at?rand=XXXXXX
URLS mit Parameter	
Ursprüngliche URL	www.google.at?query=abc
Angepasste URL	www.google.at?query=abc&rand=XXXXXX

Tabelle 1: URLs mit Parametern

Es bietet sich also die Möglichkeit, an die zu speichernde URL einen Parameter mit zufälligem Wert anzuhängen. Dadurch wird der Proxy dazu gebracht, dass der Client eine Ressource anfordert, die sich noch nicht im Cache befindet. Eine gute Möglichkeit, einen vernünftigen Zufallswert anzuhängen, ist die Verwendung eines Timestamps. In Tabelle 1 wird gezeigt, wie eine URL, die keinen oder mehrere Parameter enthält, angepasst werden muss, um dem erwähnten Verfahren zu entsprechen.

Falls der Proxy beziehungsweise der Server so konfiguriert sind, dass die Methode des angehängten Zufallsparameters abgelehnt wird, so wird die gecachte Version der Ressource ausgeliefert oder die Auslieferung komplett verweigert. Hierfür müssten der

Ursprungsserver bzw. die Proxys eine Liste besitzen, in der jede einzelne Ressource mit ihren gültigen Parametern angeführt ist.

Grundsätzlich kann davon ausgegangen werden, dass ein Proxy niemals diese Informationen besitzen kann, da er die relevanten Daten lediglich weiterreicht und nicht selbst besitzt.

Im Hinblick auf den Ursprungsserver bedeutet dies, sollte sich auf dem Server eine Grafik mit der Bezeichnung image.jpg befindet und diese Grafik an zwei verschiedenen Stellen in einem HTML-Dokument aufgerufen werden, einmal ohne Parameter und einmal mit dem Parameter *?select=true,* so müsste die zuvor genannte Liste einen Eintrag für beide dieser Aufrufe enthalten. In der Praxis sieht es allerdings so aus, dass der Inhalt der ausgelieferten Webseiten auf dem Webserver nicht ersichtlich ist.

Für die lokale Sicherung bedeutet dies, falls eine Ressource mehrmals mit unterschiedlichen Parametern vorkommt, so wird jede dieser Varianten gesondert vom Server angefordert und abgespeichert. Ein Problem, das dabei auftreten könnte, betrifft die Konsistenz der Daten. Würde das Bild *image.jpg* während des Sicherungsvorganges verändert werden, so würden beide Varianten in der fertigen Sicherung vorkommen und so eine Inkonsistenz auftreten.

Beim Evaluierungsprozess, der in *Kapitel 9 Evaluierung* dokumentiert wurde, trat dieses theoretische Problem, das durch das Hinzufügen neuer Parameter entstehen könnte, nicht auf. Bei keinem einzigen Sicherungsvorgang konnte darauf rückgeschlossen werden, dass ein Unterschied zwischen Original und Sicherung auf dem Hinzufügen eines Zufallsparameters beruhen würde.

5.4 Squid

Squid ist einer der am weitesten verbreiteten Web-Proxys und wird von Internet-Providern auf der ganzen Welt eingesetzt. Da der Kern von Webstamp das Erhalten von ungecachten

Daten darstellt, ist es erforderlich, sich mit Proxys auseinanderzusetzen. Idealerweise wird hier der bekannteste Proxy behandelt, Squid.

“Squid is a caching proxy for the Web supporting HTTP, HTTPS, FTP, and more. It reduces bandwidth and improves response times by caching and reusing frequently-requested web pages. Squid has extensive access controls and makes a great server accelerator. It runs on most available operating systems, including Windows and is licensed under the GNU GPL.” [6]

“Squid is used by hundreds of Internet Providers world-wide to provide their users with the best possible web access.” [6]

“Thousands of web-sites around the Internet use Squid to drastically increase their content delivery. Squid can reduce your server load and improve delivery speeds to clients.” [6]

Wie aus den Zitaten der Squid-Homepage hervorgeht, handelt es sich bei dieser Software um einen Proxy, der frei verfügbar ist und auf allen gängigen Betriebssystemen eingesetzt werden kann. Für alle praktisch orientierten Teile dieser Arbeit, die sich mit Caching beschäftigen und einen Proxy erforderlich machen, wurde Squid eingesetzt.

Squid ist für die Aufgabenstellung dieser Arbeit repräsentativ, da diese Software im großen Stil eingesetzt wird und somit davon ausgegangen werden kann, dass sich eine so weit verbreitete Software an die vorgegebenen Standards der Proxy-Technologie hält. Kurz gesagt bedeutet dies, dass es beim Surfen durch das Internet relativ wahrscheinlich ist, früher oder später die angeforderten Informationen von einem Squid-Proxy zu erhalten. Exakte Zahlen bezüglich des Marktanteiles konnten allerdings keine gefunden werden. Da es sich hierbei um freie Software handelt, gibt es keine offiziellen Verkaufszahlen, mit denen sich der Marktanteils erheben ließe.

5.5 Testszenario

Mit Hilfe von Squid können die zuvor erwähnten Methoden nun im Detail durchgespielt werden. Zuerst wird der Proxy so konfiguriert, dass er *.png-Ressourcen* beim ersten Aufruf cacht und diese nicht erneuert. Das bedeutet, dass jedes *.png-Bild*, nachdem es einmal im

Cache eingetragen wurde, nicht erneut angefordert wird. Die entsprechende Squid-Konfiguration muss in der Datei *Squid.config* eingetragen werden.

Codestück 4 zeigt wie die Konfiguration bezüglich der *.png-Dateien* auszusehen hat. Tabelle 2 erklärt die einzelnen Optionen, aus denen gewählt werden kann. Die Zahl 43200 gibt die Dauer in Sekunden an, bis der eigene Cache aktualisiert wird.

```
refresh_pattern -i \.(png)$ 43200 0% 432000 override-expire override-lastmod reload-into-ims ignore-reload ignore-no-cache ignore-private
```

Codestück 4: Konfiguration von Squid für PNGs

Einstellung	Beschreibung
override-expire	Setzt den Wert dieses Tags auf 0, somit wird die gecachte Ressource immer als aktuell angesehen.
override-lastmod	Ressourcen, die erst vor Kurzem verändert wurden, werden trotzdem nicht neu geladen.
reload-into-ims	Verändert die reload- oder no-cache-Anfragen des Clients in eine If-Modified-Since-Anfrage.
ignore-reload	Ignoriert die reload- und no-cache-Anfragen des Clients.
ignore-no-cache	Ignoriert die no-caching-Anfragen der Server.
ignore-no-store	Ignoriert alle Cache-control: no-store-Anfragen der Server.
ignore-private	Ignoriert alle Cache-control: private Anfragen der Server.

Tabelle 2: Auszug aus den Caching-Optionen von Squid

Nach der Konfiguration von Squid wurde eine HTML-Seite (*test.html*) erstellt, welche eine *.png-Ressource* referenziert. Dem folgenden Codestück 5 kann der entsprechende Code entnommen werden. Es handelt sich dabei um ein simples HTML-Skelett, das innerhalb des Bodys eine Grafik einbindet.

```
<html>
	<head>
	</head>
	<body>
		<img src='text.png'>
	</body>
</html>
```

Codestück 5: HTML-Code der Testseite zur Squid-Validierung

Beim Aufruf von *test.html* wurde die Ressource *test.png* nun sofort gecacht. Im nächsten Schritt wurde die Ressource *test.png* verändert und die Webseite erneut aufgerufen. Als Ergebnis wurde nicht die neue Ressource sondern die bereits gecachte ausgeliefert.

5.6 Überprüfung der Optionen

Mithilfe von Squid wurden beide Optionen getestet. Die entsprechende Konfiguration lässt sich aus Codestück 4 entnehmen.

- Option 1 (*no-cache*-Bitte)
    - Option 1 wurde von Squid nicht beachtet. Dies bedeutet, dass die Bitte, eine ungecachte Version der Ressource zu erhalten, ignoriert wurde. Diese Option ist offensichtlich nur dann sinnvoll, wenn die Kommunikation über Proxys läuft, die sich an den HTTP-Standard halten.
- Option 2 (Parameter anhängen)
    - Option 2 ermöglichte es, Squid so zu konfigurieren, dass die angeforderte Ressource noch nicht gecacht wurde. Als Ergebnis wurde, nach dem Anhängen eines Parameters an die Ressource, immer die aktuelle Ressource geliefert.

5.7 Zusammenfassung

Das grundlegende Problem beziehungsweise die Situation, welche besonderes Augenmerk von Webstamp erfordert, ist das Konzept des Cachings. Der Sinn von Webstamp steht und fällt basierend darauf, ob die gesammelten Informationen zum Zeitpunkt der Sicherung

auch aktuell waren. Um dieses Ziel garantieren zu können, wurden in diesem Kapitel zwei verschiedene Optionen besprochen, die einerseits dem Standard entsprechen und andererseits eine kreative Lösung darstellen.

Um zu garantieren, dass es sich bei den gesicherten Daten um den aktuellen und nicht etwa einen veralteten Datenbestand handelt, setzt Webstamp beide Methoden ein. Während darum gebeten wird, eine aktuelle Ressource zu erhalten, wird die Anfrage gleichzeitig durch das Hinzufügen eines Parameters verändert. Dadurch wird bei jeder Anfrage eine noch nicht ausgelieferte Ressource mit der Bitte *no-cache*, *no-store* und *must-revalidate* ausgeliefert.

6 Die Parameter-Methode

Im vorangegangenen Kapitel wurde im Allgemeinen die Methode des Anhängens von Parametern an Webseiten und deren Ressourcen besprochen, um das Erhalten gecachter Inhalte zu verhindern. In diesem Kapitel wird auf die Methode im Detail eingegangen unter Einbeziehung realer Anwendungsbeispiele.

Ziel der Analyse ist es, herauszufinden, wie sich weit verbreitete Webseiten verhalten, wenn die Server-Anfragen bezüglich der Ressourcen um Parameter erweitert werden. Interessant ist vor allem, ob es bereits ausreicht, ein einfaches Fragezeichen, das einen oder mehrere Parameter ankündigt, zu verwenden, um das Caching zu umgehen, oder ob die Verwendung von zusätzlichen Parametern erforderlich ist.

Weitere Untersuchungen betreffen die Vermutungen, ob es eventuell ausreicht, einen Parameter-Namen ohne Wert anzugeben, und wie Webseiten im Allgemeinen reagieren, falls sie standardmäßig keine zusätzlichen unbekannten oder doppelten GET-Parameter enthalten.

6.1 Die Methode im Detail

Nachdem eine Liste von zu sichernden URLs erstellt wurde, wird jede einzelne mittels der Caching-Direktive *no-cache* aufgerufen. Falls die angeforderte URL nicht erfolgreich ausgeliefert wird, so war diese URL nur noch auf einem Proxy vorhanden und nicht mehr auf dem Ursprungsserver. Kommt es jedoch zur Auslieferung, so beginnt nun die eigentliche Methode.

An die URL wird, falls sie keine Parameter enthält, ein einfaches Fragezeichen angehängt, und die Anfrage wird danach erneut gestellt. Hierbei soll vermerkt werden, dass ein einfaches Fragezeichen ausreicht, damit die gewählte Ressource als neu eingestuft wird. Sollten bereits Parameter vorhanden sein, so wird an diese Liste einfach ein weiterer Parameter angehängt, der wie folgt zusammengesetzt wird. Es wird ein zufälliger Buchstabe ermittelt, der mit dem aktuellen Timestamp verknüpft wird, als Wert wird

ebenfalls der aktuelle Timestamp verwendet. Dies bedeutet, dass für jeden generierten Parameter einer jeden Ressource ein neuer Zeitstempel verwendet wird. Ist dieser generierte Parameter bereits vorhanden, so wird erneut ein zufälliger Buchstabe mit dem aktuellen Timestamp verknüpft, bis ein Parameter gefunden wurde, der noch nicht vorhanden ist. Dieser grundlegende Ablauf wird in Abbildung 4 grafisch veranschaulicht.

Nach dem Finden eines passenden Parameters wird der HTTP-Request gestellt. Falls die URL nicht erfolgreich ausgeliefert wurde, so befand sich die Datei nur noch auf einem Proxy oder im Cache des Ursprungsservers. Ansonsten kann davon ausgegangen werden, dass es sich bei der gesicherten Datei um eine Datei handelt, die zum Sicherungszeitpunkt auch wirklich existierte und auch genauso aussah wie zu diesem Zeitpunkt.

In Abbildung 5 ist der Ablauf zur Bestimmung der Herkunft von Ressourcen in Form eines UML-Diagrammes abgebildet. Die Nummerierung von eins bis fünf zeigt den idealen Ablauf, während die Schritte 1.1 und 3.1 in einen Fehlerzustand führen, welcher innerhalb der Software selbstverständlich behandelt wird.

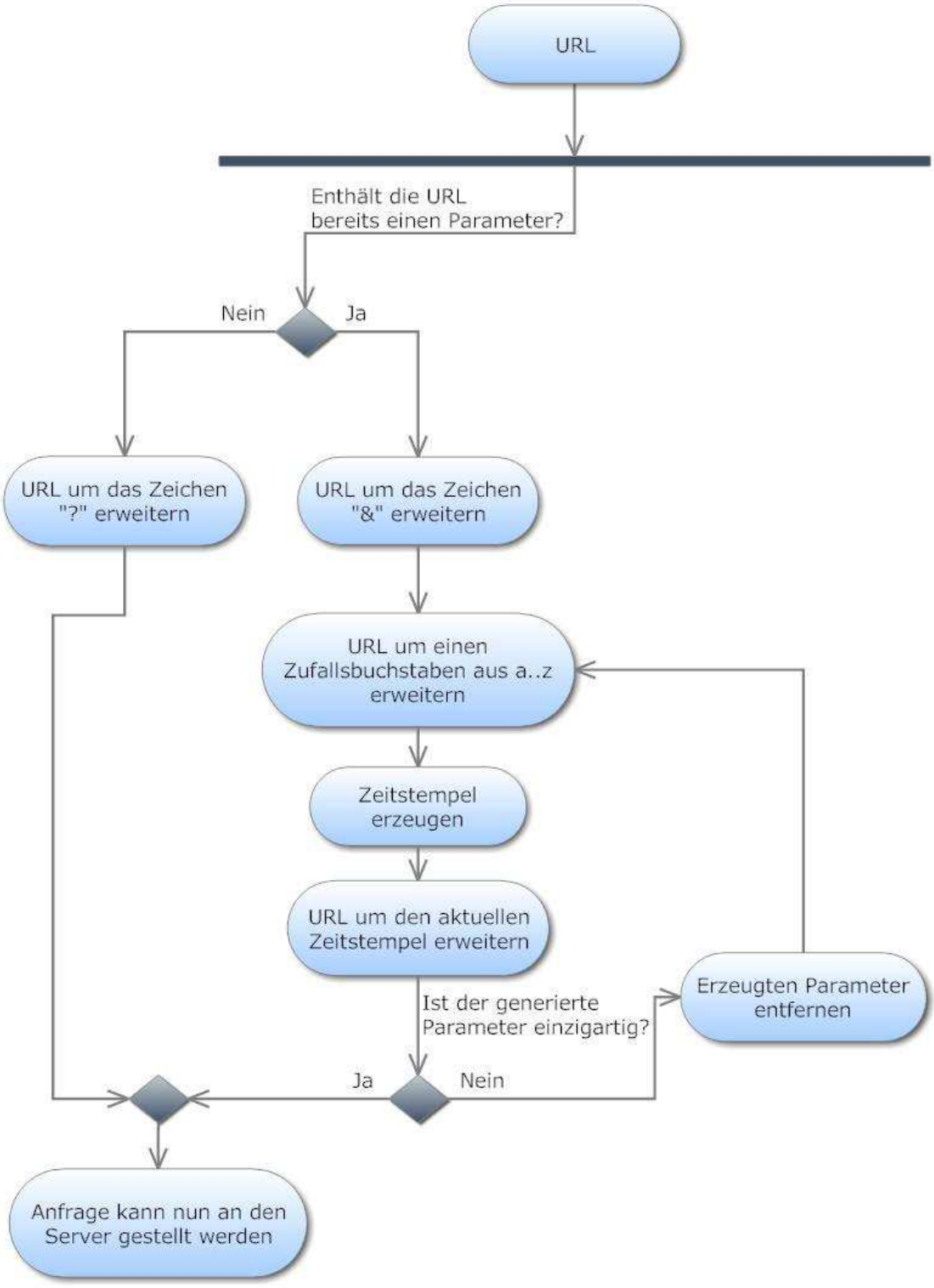

Abbildung 4: UML, Parameter-Methode

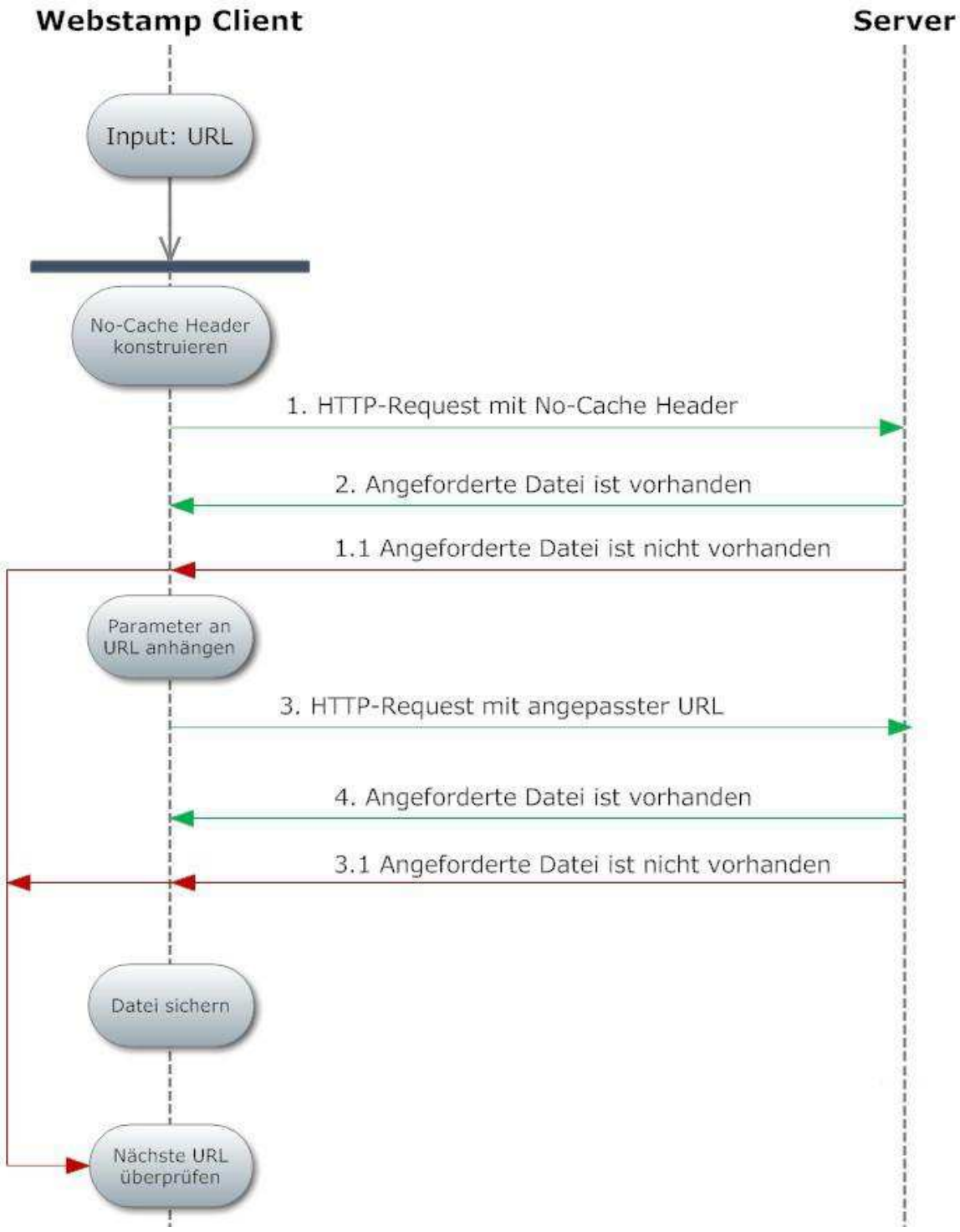

Abbildung 5: UML, Herkunft von Ressourcen bestimmen

6.2 Die Methode im Einsatz

Die in diesem Kapitel beschriebene Methode wurde an einer breiten Auswahl von Webseiten getestet. So wurden Webseiten mit einem hohen Pagerank wie Google und Amazon, aber auch Webseiten mit einem niedrigen Pagerank wie die ORF-Webseite, unter die Lupe genommen. Eine genaue Auflistung der getesteten Webseiten sowie die Definition des Pageranks finden sich im *Kapitel 9 Evaluierung* wieder.

Die Parameter-Methode an sich führte bei keinem einzigen Sicherungsvorgang zu einem Problem und wurde von jedem Webserver akzeptiert. Die Befürchtung, dass Webserver die Parameter der einzelnen Ressourcen zählen und so die Anfrage ablehnen könnten, stellte sich im Rahmen der Evaluierung als unbegründet heraus.

7 Änderungszeitpunkt von Webseiten

Zu Beginn dieser Arbeit stand die Frage im Raum, ob und wenn ja wie es möglich wäre, sicherzustellen, dass es sich bei dem Datenbestand einer zu sichernden Webseite auch wirklich um aktuelle Daten handelt. In diesem Kapitel werden die verschiedenen Möglichkeiten zur Bestimmung des letzten Änderungsdatums einer bestimmten Webseite aufgezeigt. Hierbei muss unterschieden werden, ob es sich um eine statische oder eine dynamisch generierte Webseite handelt.

7.1 Statische Webseiten

Hierbei handelt es sich um Webseiten, deren Inhalte sich nicht verändern. Der Server liefert einfach die zuvor gespeicherten HTML-Seiten unverändert an den Client aus. Dadurch ergibt sich die Möglichkeit festzustellen, wann das angeforderte Dokument das letzte Mal modifiziert wurde. Im Weiteren lässt dies die Möglichkeit zu, auf das Änderungsdatum der Webseite rückschließen zu können.

Beispielsweise kann durch die Eingabe des im folgenden Codestück 6 anfgeführten JavaScripts in die Adresszeile des Browsers das letzte Änderungsdatum der gerade angezeigten Webseite in Erfahrung gebracht werden.

```
javascript:alert(document.lastModified)
```

Codestück 6: JavaScript, Änderungszeitpunkt von Dokumenten

7.2 Dynamisch generierte Webseiten

Dynamische Webseiten nutzen Serversprachen wie beispielsweise PHP, ASP und JAVA, um aus den entsprechenden Skripten Dokumente für die Clients zu erstellen. Das Problem dabei, ist, dass eine nicht aktuelle Webseite als aktuell erscheint, da das ausgelieferte Dokument eben erst vom Server erzeugt wurde.

7.3 Verwendung in Webstamp

Gegen Mitte der Arbeit wurde festgehalten, dass die Bestimmung des letzten Änderungsdatums einer Webseite für Webstamp im Grunde nicht relevant ist, da die Hauptaufgabe der Software darin besteht, den aktuellen Zustand, wie er einem Benutzer beim Aufruf der Website angezeigt wird, festzuhalten.

Die Informationen bezüglich der letzten Änderungen können dennoch aus den Log-Dateien entnommen werden, falls der Server diese Information mit seinem HTTP-Response als Meta-Daten mitlieferte. Die Informationen befinden sich in der Datei *(Projektname)_header.log*. Würde das Projekt den Namen Google tragen, so würden sich die gesuchten Informationen in der Datei *Google_header.log* befinden.

8 Webstamp

Abbildung 6: Webstamp-Logo

Die Software, die im Rahmen dieser Arbeit entstand, wurde von mir Webstamp genannt was eine Zusammensetzung aus den Wörtern Website und Timestamp bildet. Damit soll verdeutlicht werden, um welches Themengebiet es sich bei dieser Arbeit handelt - das Sichern von Webseiten durch die Verwendung von Zeitstempeln.

Das relativ einfach strukturierte und schnell zu überschauende GUI (Graphical User Interface) ist in Abbildung 7 abgebildet. Bei der Entwicklung des GUIs wurde viel Wert darauf gelegt, dieses so einfach wie möglich zu halten, damit der Benutzer mit möglichst wenig Aufwand sein Ziel erreichen kann.

Abbildung 7: Webstamp GUI

8.1 Einführung

Die Applikation ist in der Entwicklungssprache Java geschrieben. Gründe dafür sind das bereits fundierte Wissen über diese Sprache meinerseits und, dass Java potenziell auf allen Systemen lauffähig ist. Zu Beginn der Arbeit stand die Frage im Raum, ob es Sinn machen würde, die Softwarelösung als Plugin, beispielsweise für Firefox, zu realisieren. Letztendlich wurde aber dagegen entschieden, da befürchtet wurde, dass die gewünschten Daten durch den Browser direkt oder indirekt verändert werden würden.

Unter direkte Änderungen würde die Darstellung bestimmter HTML-Elemente fallen, die Abstände zwischen den einzelnen Elementen oder die Darstellung grafischer Elemente wie Linien wären Beispiele für dieses Problem.

Unter indirekte Änderungen fällt die Situation, in denen der Webserver, der die gewünschten Daten zur Verfügung stellt, den HTTP-Request des Clients nach dem Typ und jener Version des Browsers durchsucht, der die Anfrage stellte, und danach eine maßgeschneiderte Antwort liefert. Diese Situation tritt sehr häufig auf, da zwischen den Browsern und ihren Versionen gravierende Unterschiede bestehen. Es ist sogar üblich,

unterschiedliche Stylesheets oder Codeblöcke auszuliefern. Meist handelt es sich dabei um Hacks, welche Features höherer Versionen bzw. anderer Browser realisieren.

8.2 Konfiguration: User-Agent

Im vorangegangenen Abschnitt *7.1 Einführung* wurde erklärt, weshalb Webstamp nicht als Browser-Plugin sondern als Java-Anwendung realisiert wurde. Diese Entwicklungsentscheidung ermöglichte es, ein weiteres Feature zu realisieren, das Durchführen von Browser-spezifischen Sicherungen. Hierfür befindet sich im Root-Verzeichnis der Anwendung eine Textdatei mit dem Namen c*onfig.txt* deren Inhalt in Codestück 7 angegeben ist.

```
#In dieser Konfigurations-Datei wird der User-Agent angegeben welcher bei jedem HTTP-Request angegeben wird.

#Zeilen welche mit einem '#' Zeichen beginnen sind Kommentare.

#Der User Agent kann in einer ganzen Zeile oder zeilenweise angegeben werden, Leerzeichen werden vom System selbstständig hinzugefügt falls der User-Agent über mehrere Zeilen angegeben wird.


#User Agent

Mozilla/6.0

(Macintosh; I; Intel Mac OS X 11_7_9; de-LI; rv:1.9b4)

Gecko/2012010317

Firefox/10.0a4


#Andere User-Agents

#Chrome: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/535.2 (KHTML, like Gecko) Chrome/18.6.872.0 Safari/535.2 UNTRUSTED/1.0 3gpp-gba UNTRUSTED/1.0

#Firefox: Mozilla/6.0 (Macintosh; I; Intel Mac OS X 11_7_9; de-LI; rv:1.9b4) Gecko/2012010317 Firefox/10.0a4

#IE: Mozilla/5.0 (compatible; MSIE 10.6; Windows NT 6.1; Trident/5.0; InfoPath.2; SLCC1; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729; .NET CLR 2.0.50727) 3gpp-gba UNTRUSTED/1.0
```

Codestück 7: Webstamp-UserAgent-Konfiguration

Die Aufgabe eines User-Agents ist es, eine Aufgabe im Auftrag des Benutzers auszuführen. Beispielsweise wäre ein Programm zum Lesen von E-Mails ein Mail-User-Agent. [17] Für einen Webbrowser bedeutet das, dass der User-Agent Informationen über das OS und den verwendeten Browser ausliefert.

In der Datei *config.txt* kann der User-Agent in einer einzelnen Zeile oder zeilenweise angegeben werden. Bei der Durchführung der Sicherung liest Webstamp den User-Agent aus dieser Datei ein und übermittelt ihn bei jeder Anfrage an den Server. Die Wahl des User-Agents kann sich gravierend auf die Sicherung auswirken, so könnte beispielsweise der User-Agent eines mobilen Gerätes angegeben werden. Als Antwort würde hierbei vom Server das Stylesheet für mobile Endgeräte geliefert werden, falls vorhanden.

8.3 Dateien

Alle von Webstamp benötigten Dateien und Ordner werden in der folgenden Tabelle 3 aufgeführt und erklärt.

Datei- und Ordnerstrukturen von Webstamp	
webstamp.jar	In dieser Datei befindet sich die eigentliche Programmlogik. Mittels einer Konsole kann diese Datei angesprochen und mit Parametern aufgerufen werden.
webstamp_GUI.jar	Aus Gründen der Benutzerfreundlichkeit wurde für Webstamp ein eigenes GUI (Graphical User Interface) entwickelt, dass es dem Benutzer erlaubt, alle Einstellungen vorzunehmen. Nach dem Ausfüllen der benötigten Felder startet diese Datei die Datei webstamp.jar mit den zuvor eingegebenen Parametern.

config.txt	In dieser Datei können die Einstellungen für den gewünschten Browser und die gewünschte Version angegeben werden die bei der Sicherung verwendet werden.
img/webstamp.jpg	Das Bild, das bei der Verwendung des About-Menüs in der grafischen Benutzeroberfläche angezeigt wird.

Tabelle 3: Datei- und Ordnerstrukturen von Webstamp

8.3.1 Konsolenbefehle

In diesem Kapitel werden die Konsolenbefehle *store* und *verify* erörtert.

- Der *Store*-Befehl

Der Zweck des *Store* Befehls ist die Speicherung einer Webseite. Zur korrekten Ausführung werden fünf weitere Parameter benötigt. Codestück 8 zeigt beispielhaft, wie dieser Befehl benutzt wird, Tabelle 4 stellt eine Liste aller möglichen Parameter zur Verfügung.

```
-c store -s C:/downloads –url http://www.orf.at –name orf –fetchExternalDomains true –fetchHTMLScripts false
```

Codestück 8: Webstamp Store-Befehl

Der Store-Befehl mit den möglichen Parametern		Default-Wert
-url	Die URL der zu sichernden Webseite.	Nicht vorhanden.
-dir	Speicherort auf der Festplatte, wo die Sicherung abgelegt werden soll.	Nicht vorhanden.
-name	Projektordner, der am Speicherort für die Sicherung angelegt wird.	Nicht vorhanden.
-fetchExternalDomains	Gibt an, ob Inhalte der Webseite, die sich nicht auf ihrer Domain befinden, gesichert werden sollen. Gültige Werte sind true und false. Die Verwendung von true wird empfohlen!	True
-fetchHTMLScripts	Gibt an, ob HTML-Inhalte aus JavaScript- und CSS-Dateien extrahiert werden sollen. Gültige Werte sind true und false. Die Verwendung von false wird empfohlen.	False

Tabelle 4: Webstamp Store-Befehl

Nach der erfolgreichen Ausführung des *Store*-Befehls werden einige Dateien sowie entsprechende Ordnerstrukturen erstellt. Tabelle 5 gibt Aufschluss darüber, wie diese konkret aussehen.

Ordner- und Dateistruktur einer Sicherung	
C:\Downloads\orf	Hier befindet sich die forensische Kopie der Webseite.
C:\Downloads\orf_live	Hier befindet sich die für das lokale Dateisystem angepasste Kopie der Webseite, die Standalone-Version.
C:\Downloads\orf_checksum.log	Eine Textdatei, die Namen und Hash-Werte aller Dateien enthält.
C:\Downloads\orf_default.log	Eine Log-Datei, die alle Informationen enthält, die dem Benutzer während des Sicherungsprozesses angezeigt wurden, allerdings mit mehr Details.
C:\Downloads\orf_filename.log	Eine Log-Datei, die angibt, wann welche Dateien umbenannt werden mussten, aufgrund der Problematik, die in Kapitel 8.5.1 Kürzen zu langer URLs behandelt wurde.
C:\Downloads\orf_header.log	Enthält für jede gesicherte Datei den HTTP-Request und den HTTP-Response.
C:\Downloads\orf_certificate.log	Dies ist das Zertifikat des Zeitstempeldienstes, die für den Hash-Wert der Datei orf_checksum.log erstellt wurde.

Tabelle 5 : Webstamp Ordnerstrukturen und Dateinamen

- Der *Verify*-Befehl

Der Zweck des *Verify*-Befehles ist es, feststellen zu können, ob eine Sicherung zu einem späteren Zeitpunkt verändert wurde. Zur korrekten Ausführung wird ein weiterer Parameter benötigt. Codestück 9 zeigt beispielhaft, wie dieser Befehl aussieht. Tabelle 6 erklärt den benötigten Parameter *-dir*.

```
-c verify -dir C:/downloads
```

Codestück 9: Webstamp Verify-Befehl

Der Verify-Befehl und sein Parameter		**Default-Wert**
-dir	Der Ordner, in dem sich die Sicherung befindet.	Nicht vorhanden.

Tabelle 6: Webstamp Verify-Befehl

- Der *Help*-Befehl

Der *Help*-Befehl liefert dem Benutzer dieses Kapitels einen Einblick in die Verwendung der Software. Codestück 10 zeigt beispielhaft, wie dieser Befehl aussieht.

```
-c help
```

Codestück 10: Webstamp Help-Befehl

8.4 Funktionalität

Webstamp stellt drei grundlegende Funktionen zur Verfügung:

- Webseite sichern
    - Ermöglicht es, durch die Eingabe einer URL und das Festlegen von zwei Parametern den Sicherungsvorgang zu starten.
- Sicherung verifizieren

    - Durch die Angabe des Projektordners wird die darin enthaltene Sicherung überprüft. Hierbei wird festgestellt, ob die gesicherten Daten unverändert sind und ob Daten hinzugefügt oder entfernt wurden.

- Hilfe
    - Die Hilfe liefert alle Informationen, die für die Benutzung von Webstamp erforderlich sind.

8.4.1 Transferieren der Sicherung

Eine wesentliche Funktion der Applikation ist die Möglichkeit, eine gespeicherte Webseite auf anderen Rechnern in anderen Ordnerstrukturen wiedergeben zu können. Auf den ersten Blick scheint diese Anforderung trivial zu sein, bei näherer Betrachtung ergibt sich allerdings ein durchaus problematisches Szenario.

Nehmen wir als Beispiel die fiktive URL *www.example.com* ein Stylesheet referenziert unter *www.example.site.com/styles/style.css*. In diesem Stylesheet werden nun zwei Grafiken referenziert.

Die erste Grafik hat den relativen Pfad *../images/image01.jpg* und die zweite Grafik den absoluten Pfad *www.example.site.com/styles/newimages/image02.jpg*.

Der Pfad der zweiten Grafik verweist absolut auf eine Ressource im Web. In der lokalen Version der Webseite müsste dieser geändert werden, da ansonsten eine Ressource aus dem Web referenziert werden würde. Hierbei ergeben sich zwei Möglichkeiten.

Erstens könnte man den Pfad komplett auf das lokale Dateisystem abändern.

Nehmen wir an, die Webseite würde im Ordner *C:\Downloads* abgelegt, so würde sich hierbei der Pfad *C:\Downloads\www.example.site.com\styles\newimages\image02.jpg* ergeben.

Zweitens könnte man den Verweis auf die Domain einfach entfernen.

Dies würde den relativen Pfad *styles\newimages\image02.jpg* ergeben. Das große Problem hierbei ist, dass Verweise in Stylesheets immer relativ zu dem Stylesheet sind, in dem sie definiert wurden, und nicht zu dem Bezugspunkt, wo sie verwendet werden. Dies würde somit den falschen Pfad ergeben:

C:\Downloads\www.example.site.com\styles\styles\newimages\image02.jpg. Um einen korrekten Pfad zu erhalten, müsste zusätzlich zum Speicherort der Pfad *styles* entfernt werden. Hierbei müsste man wissen, in welchem Stylesheet die Ressource referenziert wurde. Bei der Abänderung auf einen relativen Pfad müssten sämtliche vorangegangenen Ordner entfernt werden. Dies wäre möglicherweise realisierbar, aber mit erheblichem Aufwand verbunden.

Die einfachste Lösung für dieses Problem wäre die Verwendung der ersten Variante. Allerdings ist es hierbei erforderlich, die absoluten Pfade in allen Dateien anzupassen, sollte die Sicherung verschoben werden. Dies ist weiterhin problematisch, da die gesicherten Daten dabei verändert werden würden, es sei denn man würde nach einem Datentransfer nur mit temporären Dateien, die aus der eigentlichen Sicherung generiert werden, arbeiten.

Letztendlich wurde diese Problematik gelöst, indem zwei verschiedene Versionen einer Webseite gesichert werden. Die erste Version entspricht komplett jener des Webservers ohne irgendwelche Veränderungen der Pfade. Die Darstellung an sich ist allerdings mit einigen Problemen behaftet, falls z. B. fehlerhafte Pfadinformationen enthalten sind. Kopiert man diese Version in das Root-Verzeichnis eines Webservers, so verhält sich diese Kopie ähnlich wie die Originalversion. Um zu vermeiden, dass absolute Ressourcen angezeigt werden, ist es erforderlich, den Browser in den Offline-Modus zu versetzen.

Um nicht zwingend an einen Webserver gebunden zu sein und dem Problem der absoluten Ressourcen entgegenzuwirken, entspricht die zweite Version dem lokalen Dateisystem und passt alle Pfade für selbiges an. Diese Version beinhaltet somit keine 1:1-Kopie, es wird vielmehr versucht, die Darstellung der Webseite auf dem Ursprungsserver zu kopieren.

8.4.2 Feststellen von Veränderungen

Eine wesentliche Funktion der Applikation ist die Möglichkeit, gespeicherte Webseiten auf andere Systeme zu transferieren. Ein grundlegendes Problem, das sich hier ergibt, ist die Möglichkeit, dass die gesicherten Daten auf ihrem Weg zum neuen Speicherort verändert werden könnten. Dies kann fatale Folgen haben, sollten – egal ob wissentlich oder unwissentlich – veränderte Daten zur Beweisführung herangezogen werden.

Aus diesem Grund wird im Zuge des Sicherungsvorgangs eine Liste erstellt, die jede einzelne Datei mit ihrem entsprechenden Hash-Wert enthält. Um die Integrität dieser Liste sicherstellen zu können, wird diese, nachdem sie gefüllt wurde, an einen Zeitstempeldienst übermittelt, der mit einer Signatur der entsprechenden Liste antwortet. Diese Signatur kann als Zertifikat betrachtet werden. Zu einem späteren Zeitpunkt kann so mithilfe des Zeitstempeldienstes und der Signatur beziehungsweise des Zertifikats festgestellt werden, ob die Liste unverändert ist.

Es sollte noch erwähnt, werden dass nicht die Liste selbst, sondern ein Hash-Wert dieser Liste, an den Zeitstempeldienst übermittelt wird. Dies hat den Grund, dass es niemanden etwas angeht, welche Informationen sich in dieser Liste befinden. Da es nicht möglich ist, von einem Hash-Wert auf die Original-Daten zu schließen, können auf diese Art sensible Daten über unsichere Kanäle, in diesem Fall das Internet, übertragen werden.

Mehr Informationen diesbezüglich finden sich im *Kapitel 10.2 Zeitstempeldienste* wieder.

8.5 Überlegungen zur Programmierung

In diesem Kapitel werden wichtige Entscheidungen, die im Rahmen der Programmierung der Applikation getroffen wurden, genauer erörtert.

8.5.1 Kürzen zu langer URLs

In allen gängigen Dateisystemen wie NTFS oder ext3 ist die Länge des Dateinamens auf maximal 256 Zeichen begrenzt. Die Länge einer URL kann über diese Einschränkung weit hinausgehen, der Internet Explorer erlaubt zum Beispiel bis zu 2048 Zeichen. [18]

Eine 1:1 – Abbildung von URLs auf das lokale Dateisystem ist somit durchaus mit Problemen behaftet, die auf zwei verschiedene Arten gelöst werden könnten.

Der erste Ansatz behandelt die Möglichkeiten, eine zu lange URL zu kürzen, indem ein Teil davon abgeschnitten wird. Dabei wird die Frage gestellt, ob links oder rechts gekürzt werden soll. Da HTML von links nach rechts geparst wird, kann nichts vom linken Teil entfernt werden, falls der Dateiname zu lang für das Dateisystem ist. Aus diesem Grund muss bei diesem Ansatz der überschüssige rechte Teil abgeschnitten werden. Hierfür wurde eine Maximallänge von 232 Zeichen eingeführt, die in Tabelle 7 erklärt wird. In der Praxis konnten mit dieser Methode sehr gute Ergebnisse erzielt werden. Ein Problem, das dabei jederzeit auftreten kann, ist, dass zwei oder mehr URLs nach der definierten Maximallänge immer noch ident sind. In diesem Fall würden alle gleichen URLs mit dem Informationsgehalt der Letzten überschrieben.

Die zweite Herangehensweise an diese Problematik ist die Umbenennung zu langer URLs. Auf den ersten Blick wirkt diese Lösung relativ trivial, da es ja lediglich erforderlich ist, eine Liste aller gefundenen URLs zu erstellen und beim Auftreten einer zu langen URL einen passenden neuen Dateinamen zu wählen, der noch nicht existiert.

Das große Problem bei dieser Lösung ist, dass sehr viele JavaScripte URLs dynamisch zusammenbauen. Da es nicht möglich ist, unvollständige URLs in Skripten ebenfalls neu zu benennen, weil Fehler hierbei sehr wahrscheinlich wären, würde jeder Verweis auf diese Ressourcen schlicht und einfach ungültig sein.

Betrachtet man die beiden Lösungswege, ist es objektiv gesehen besser, den ersten zu wählen, da hier mit geringerem Aufwand bessere Ergebnisse erzielt werden können. Somit muss die Maximallänge, nach der eine URL abgeschnitten wird, noch definiert werden.

Eine URL, die im Pfad *C:\Dokumente\Websites\jku.at* gespeichert wurde, kann bei einer Maximallänge von 255 maximal 203 Zeichen enthalten. Für sämtliche in der Praxis getesteten Webseiten war dies ausreichend, selbst für die Google-Suchergebnisse.

Einen Überblick über diese Einschränkung gibt Tabelle 7.

Maximallänge = 255 – 23 (232)

Maximale Zeichenlänge	**255**
Zum Umgehen der Caching-Mechanismen muss die URL erweitert werden, dies erfordert eines der beiden Zeichen **(?, &)** sowie eine Variable **(x)** mit Zeitstempel und ein **(=)** gefolgt von einem Zeitstempel. Die Länge von 23 ergibt sich daraus, dass ein Zeitstempel zur Zeit 10 Stellen hat und sich das in den nächsten 250 Jahren nicht ändern wird. Bsp.: (?x481608000=481608000)	23

Tabelle 7: Festlegen der Maximallänge von URLs

Um zu einem späteren Zeitpunkt nachvollziehen zu können, welche URLs gekürzt wurden, wird eine Log-Datei benutzt. Darin enthalten sind das Datum und die Uhrzeit der Kürzung sowie die Original-URL und die gekürzte.

8.5.2 Umbenennung von Dateien

Bei der Speicherung von Ressourcen ist es gelegentlich erforderlich, den eigentlichen Dateinamen zu verändern, da bestimmte Zeichen, die im Web Verwendung finden, nicht in einem lokalen Dateisystem benutzt werden können.

Als Beispiel dienen folgende Ressourcen.

- *http://example.com/style.css?var=1*

    - Liefert ein Stylesheet mit blauer Schriftfarbe.
- *http://example.com/style.css?var=2*
    - Liefert ein Stylesheet mit roter Schriftfarbe.

Das Stylesheet, das sich hinter dem Dateinamen *style.css* befindet, ist abhängig vom Parameter *var*. Je nachdem, welchen Wert dieser Parameter annimmt, wird ein unterschiedliches Stylesheet ausgeliefert. Eine einfache Speicherung von *style.css* (Bsp.: *C:\Downloads\projekt\example.com\style.css*) würde somit zu falschen Inhalten führen, ganz abgesehen davon, dass die Verweise auf diese Ressource ungültig wären.

Es ist somit erforderlich, die Parameter in den lokalen Dateinamen miteinzubeziehen, um Eindeutigkeit zu garantieren. Hierbei kommt es aber zu dem Problem, dass URLs Zeichen enthalten können, die in einem Dateisystem für besondere Zwecke reserviert wurden. Um dieses Problem zu lösen, werden alle reservierten Zeichen durch das Unterstrich-Zeichen „_“ ersetzt. Tabelle 8 enthält alle ungültigen Sonderzeichen mit ihrer Bedeutung.

Reserviertes Zeichen	**Bedeutung**	**Kann in einer URL vorkommen ?**
<	Less than	Ja
>	Greater than	Ja
:	Colon	Ja
„	Double quote	Ja
/	Forward slash	Nein, wird zum Angeben von Ordnerstrukturen (echten und virtuellen) verwendet.
\	Backslash	Ja
|	Vertical var	Ja
?	Question mark	Ja

*	Asterisk	Ja

Tabelle 8: Nicht erlaubte Zeichen für Dateinamen in Windows Systemen

In dem gewählten Beispiel würde dies bedeuten, dass folgende Dateien angelegt werden würden:

- *C:\Downloads\projekt\example.com\style.css_var=1*
- *C:\Downloads\projekt\example.com\style.css_var=2*

Dies macht es erforderlich, dass alle Verweise auf diese Ressourcen ebenfalls umbenannt werden müssen.

Eine Umbenennung einzelner Dateien ist ebenfalls nötig, falls die zu sichernden Daten keine Dateiendungen aufweisen. Beispielsweise könnte dieser Link vorhanden sein: *http://www.example.com/example,* der lokal wie folgt abgebildet wird:

C:\Downloads\project\example.com\example.

Nun tritt ein Problem auf, falls sich eine weitere Ressource unter *http://www.example.com/example/image1.jpg* befindet, da die Ressource *image1.jpg* im Ordner *C:\Downloads\project\example.com\example* abgelegt werden muss. Da aber bereits eine Datei mit dem Namen *example* existiert, kann es keinen Ordner geben, der denselben Namen wie eine bereits existierende Datei hat. Aus diesem Grund erhält jede Ressource ohne Dateinamen die Endung *.file*.

Die forensische Kopie sowie die Standalone-Kopie der Webseite weichen in diesen Punkten von der Version des Servers ab. All diese Änderungen werden in der Log-Datei *(Projektname)_filename.log* festgehalten.

8.5.3 Herkunft von Ressourcen bestimmen (Server/Proxy)

Kern der Applikation ist die Sicherung von Webseiten in der Form, wie sie auf den Ursprungsservern vorhanden sind. Um dies zu erreichen, müssen nebst dem Setzen entsprechender Caching-Direktiven die URLs, die auf die Ressourcen verweisen, verändert

werden, und zwar durch das Hinzufügen von Parametern. Die Überprüfung, ob eine Ressource existiert oder auch nicht, kann in verschiedenen Phasen erfolgen, welche alle ihre Vor- und Nachteile aufweisen. Im Folgenden werden einige Möglichkeiten durchgespielt, und es wird eine Methode ausgewählt.

8.5.3.1 Überprüfung beim Sicherungsvorgang

Hierbei wird jede URL, nachdem sie aus der Webseite extrahiert und als relevant befunden wurde, vor dem eigentlichen Download-Vorgang verändert.

Der klare Vorteil ist die Kürze des gesamten Vorgangs, da jede Ressource nur einmal behandelt werden muss. Ein weiterer Vorteil ist die Wahrung eines validen Datenbestandes, da es niemals zu einer Sicherung von auf dem Ursprungsserver nicht mehr vorhandenen Daten kommt.

Ein Nachteil ist die Abschwächung der späteren Nachvollziehbarkeit der Sicherung. Damit wird das Problem behandelt, dass es nach dem Sicherungsvorgang nicht mehr möglich ist festzustellen, ob eine Ressource grundsätzlich nicht existiert hat, weder auf einem Proxy- noch auf dem Ursprungsserver, oder ob die Ressource zwar auf einem Proxy-Server, aber nicht mehr auf dem Ursprungsserver, existierte.

8.5.3.2 Überprüfung nach dem Sicherungsvorgang

Bei der Überprüfung nach dem eigentlichen Sicherungsvorgang wird die Webseite zuerst gespeichert und eine Liste aller gesicherten Ressourcen angelegt. Nach der Sicherung wird jede auf dem Datenspeicher vorhandene Ressource auf ihre Existenz auf dem Ursprungsserver hin überprüft. Kann die entsprechende Information so nicht mehr gefunden werden, so wird sie aus dem Datenbestand entfernt.

Der Vorteil daraus ist die vollständige Nachvollziehbarkeit, ab wann und wo eine Ressource zum Zeitpunkt der Sicherung existierte, oder ob sie nirgends existierte.

Ein Nachteil ist die längere Dauer der Sicherung, da jede Ressource zweimal behandelt werden muss. Zusätzlich entsteht ein Overhead für die Zuordnung von URLs zu den

entsprechenden Dateipfaden, die spätestens bei der Zuordnung zu abgeschnittenen URLs an nicht zu unterschätzender Komplexität gewinnt.

8.5.3.3 Die gewählte Methode

Das Mittel der Wahl liegt zwischen diesen beiden Methoden.

Jede Ressource wird, basierend auf ihrer Ursprungs-URL, auf ihre Existenz hin überprüft. Kann die Ressource nicht gefunden werden, so ist es eindeutig, dass die Ressource weder auf einem Proxy noch auf dem Ursprungsserver existiert. Wird sie hingegen gefunden, so muss die URL verändert werden und eine erneute Überprüfung erfolgen. Ist diese positiv, die Ressource wurde gefunden, so existiert die Ressource tatsächlich auf dem Ursprungsserver. Ansonsten existiert lediglich eine Kopie auf einem Proxy-Server.

Die hierbei gewonnenen Erkenntnisse können nun dafür genutzt werden, um zu entscheiden, ob eine Ressource ihren Weg auf den Datenspeicher findet oder auch nicht. Ist sie auf dem Ursprungsserver vorhanden, so wird sie gesichert, ansonsten nicht.

8.5.4 Überlegungen zur Extrahierung von URLs

Idealerweise würden bei der Sicherung einer Webseite alle relevanten URLs bezüglich der Darstellung extrahiert werden. Ein großes Problem hierbei ist es, dass nicht festgestellt werden kann, ob ein Link relevant ist oder nicht.

Eine sehr einfache Methode, die hierfür implementiert wurde, ist das Ignorieren von URLs, die sich auf andere Domains beziehen als auf die der zu sichernden Webseite. Diese Methode ist optional, da zahlreiche Webseiten heutzutage ihre relevanten Inhalte von verschiedenen Domains beziehen. Beispielsweise könnte die zu sichernde Webseite unter der Domain *example.com* erreichbar sein und Bilder von der Domain *images.example.com* enthalten. Eine potenzielle Optimierung der Software wäre, dass Subdomains nicht als domainfremde Inhalte zählen.

Um diesem Problem, zumindest größtenteils, Herr zu werden, werden keine HTML-Inhalte gesichert, abgesehen von der Start-URL und Inhalten in IFrames und Framesets.

Problematisch wird dieses Vorgehen bei der Extrahierung von URLs aus JavaScripten und CSS-Dateien, da es nicht möglich ist, zu entscheiden, ob sich eine HTML-Ressource auf ein IFrame bzw. ein Frameset bezieht oder nicht. Dies ist ein Problem, da JavaScript auch innerhalb eines IFrames Inhalte nachladen kann. Man steht hier also vor der Wahl, alle gefundenen HTML-Inhalte zu sichern, um keine möglicherweise relevanten Inhalte zu verlieren, oder HTML-Inhalte aus JS- und CSS-Dateien grundsätzlich zu ignorieren.

Am Beispiel einer Google Suche werden diese beiden Optionen kurz durchgespielt.

- Option 1 (Alles sichern)
    - Das Sichern nahm über 20 Minuten in Anspruch, da sämtliche verlinkten Webseiten aus der Google-Suche gespeichert wurden. Der Grund dafür ist, dass die URLs zu den Suchergebnissen in einem JavaScript enthalten sind.
- Option 2 (Keine HTML-Inhalte aus JS/CSS)
    - Nach knapp 1-2 Minuten war die Sicherung abgeschlossen und vollständig.

Basierend auf diesem Beispiel wäre es nun verlockend, Option 2 zu wählen, allerdings kann nicht davon ausgegangen werden, dass HTML-Inhalte, die in JS/CSS-Dateien verlinkt werden, grundsätzlich nicht relevant sind. Die einzig sinnvolle Lösung für dieses Problem scheint zu sein, die Inhalte immer zu sichern oder die Wahl dem Benutzer zu überlassen. Letztendlich fiel die Entscheidung, die Wahl dem Benutzer zu überlassen. Eine entsprechende Funktion wurde innerhalb von Webstamp implementiert.

8.6 Ablauf des Sicherungsprozesses

Der grundlegende Ablauf des Sicherungsprozesses eines Dokumentes, der über seine URL identifiziert wird, findet sich in Abbildung 8 wieder.

Die wesentlichen Eingabeparameter sind die URL der gewünschten Webseite sowie zwei Optionen.

- Die erste Option legt fest, ob Inhalte, die sich unter einer anderen Domain als der ursprünglichen URL befinden, gesichert werden dürfen.

- Die zweite Option legt fest, ob URLs aus JavaScript-Dateien entnommen werden dürfen.

Nach der Übermittlung der Eingangsparameter wird das unter der URL befindliche Dokument gesichert und nach weiteren URLs durchsucht. Wenn hier von der Sicherung eines Dokumentes gesprochen wird, so ist damit immer der HTML-Quellcode gemeint. Das Durchsuchen nach weiteren URLs hat den Zweck, die Elemente des aktuellen Dokumentes, wie beispielsweise Bilder, ebenfalls zu sichern.

Jede gefundene URL wird zuerst dahingehend überprüft, ob sie mit Option 1 konform ist. Dies bedeutet, sollte sich die neue URL auf einer anderen Domain befinden als die URL aus dem Ausgangsparameter, so wird diese, falls Option 1 aktiviert wurde, nicht gesichert.

Jede neue Datei wird, falls es sich um eine HTML-, CSS- oder JS-Datei handelt, ebenfalls nach neuen URLs durchsucht. Sollte Option 2 aktiviert sein, so werden alle URLs, die als Inhalt ein HTML-Dokument liefern, ignoriert.

Falls keine neuen URLs mehr gefunden werden können und alle gefundenen abgearbeitet wurden, so wird über die gesamte Sicherung eine elektronische Signatur gelegt.

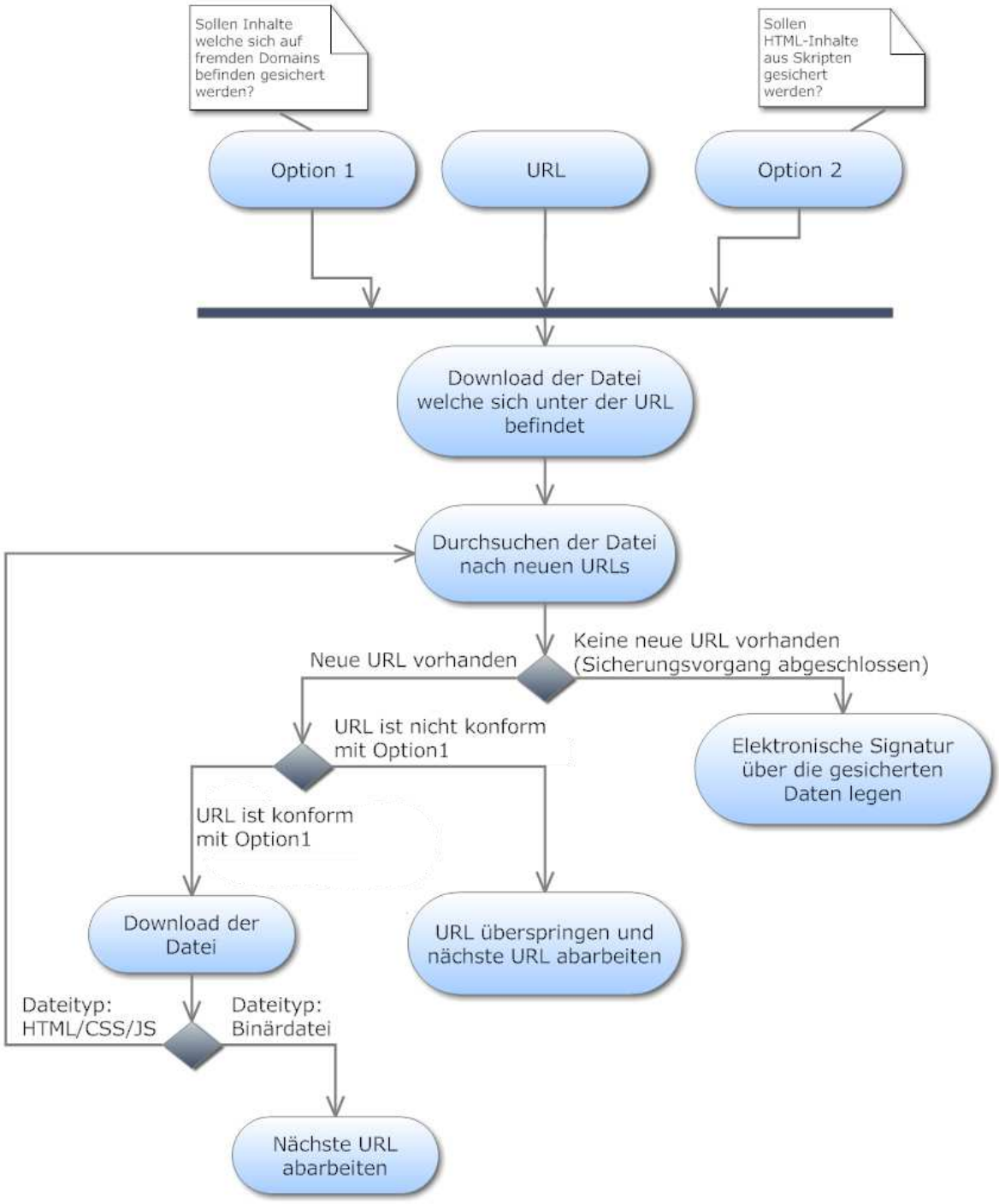

Abbildung 8: Webstamp, Ablauf des Sicherungsprozesses

9 Evaluierung

In diesem Kapitel findet sich eine Auflistung von Webseiten, die mit Webstamp testweise gesichert wurden. Diese Liste dient dazu, festzustellen zu können, ob die Funktionalität zur Speicherung von Webseiten ausreichend gewährleistet ist.

Jede Webseite wurde in eine Kategorie (Groß, Klein) eingeteilt, und es wurde festgehalten, ob es sich dabei um eine reine HTML-Webseite handelt oder ob AJAX enthalten ist.

Anschließend wurde die Fehlerrate berechnet. Hierfür wurde der offizielle W3C-Validator herangezogen, der unter der URL *http://validator.w3.org* zu erreichen ist. Die Fehlerrate dient dazu, einen Eindruck davon zu erhalten, wie Webstamp auf fehlerhafte Webseiten reagiert. Im Gegensatz zu Programmiersprachen lässt HTML sehr viele Fehler durchgehen. Dies führt zu zahlreichen Webseiten, die augenscheinlich funktionieren, aber im Grunde fehlerhaft sind.

Die Evaluierung im Hinblick auf fehlerhafte Webseiten ist sehr wichtig, da es sich bei HTML um eine Sprache handelt, die von links her gelesen wird. Können die vorangegangenen Elemente nicht verstanden werden, da sie fehlerhaft sind, so bekommt der Parser ein Problem, sprich er kann das Dokument nicht verarbeiten.

Die erlangten Erkenntnisse wurden in diesem Kapitel festgehalten.

9.1 JavaScript-Problem

Ein Problem, das sich durch die gesamte Evaluierung zog, war das Vorhandensein unvollständiger URLs. Falls eine URL nicht vollständig innerhalb des Quellcodes auftritt, so kann sie nicht extrahiert werden. Die meisten Probleme, die im Rahmen der Evaluierung auftraten, konnten auf diese Situation zurückgeführt werden.

9.2 Pagerank

Um die relative Wichtigkeit einer Webseite bestimmen zu können, wird der Pagerank-Algorithmus eingesetzt. Dabei handelt es sich um eine Methode um die Rangfolge zwischen jeder einzelnen Webseite zu berechnen, basierend auf dem Graphen des World Wide Webs. [11]

„The rank assigned to a document is calculated from the ranks of documents citing it. In addition, the rank of a document is calculated from a constant representing the probability that a browser through the database will randomly jump to the document.“ [12]

Abbildung 9 verdeutlicht wie der Pagerank entsteht. Der Pagerank kann alle ganzen Zahlen von 0 bis 10 einnehmen, wobei 10 der höchste und somit beste Wert ist.

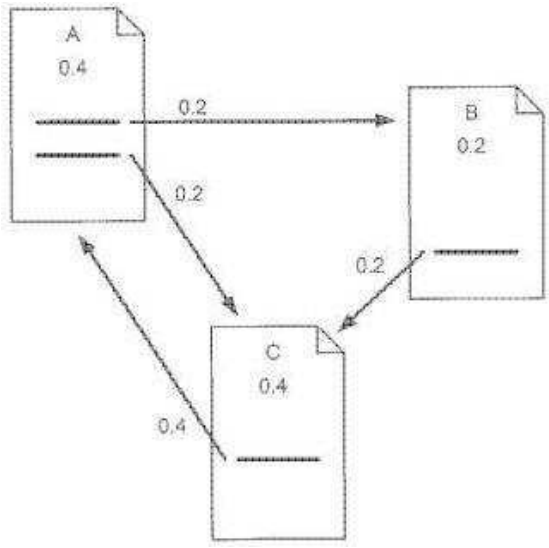

Abbildung 9: Pagerank [Pagerank]

Bei der Auswahl der Webseiten für die Evaluierung wurde darauf geachtet, Webseiten mit unterschiedlichen Pagerank-Werten zu verwenden.

9.3 Große Webseiten

Alle Webseiten in dieser Kategorie besitzen einen Google-Pagerank von mindestens 7. Abbildung 10 bis Abbildung 19 beinhalten die Ergebnisse des Evaluierungsprozesses bezüglich großer Webseiten.

9.3.1 CNN

Informationstabelle	Internationale Nachrichtenseite http://edition.cnn.com (PR: 8)
HTML-Fehlerrate	88 Fehler, 26 Warnungen
Reines HTML	Nein
Ajax	Ja
Probleme	Im oberen Bereich der Webseite wird durch ein JavaScript zufällig bestimmt, welche Grafik angezeigt wird. Es handelt sich hierbei um die Weltkugel, mit Sicht auf Amerika, Europa oder den asiatischen Raum. Da dieses Bild durch ein JavaScript zusammengesetzt wird kann nur jene Grafik gesichert werden, die zum Zeitpunkt der Sicherung zufällig ausgewählt wurde. Im unteren Bereich der Webseite wird ein JavaScript benutzt um Informationen über das Wetter auszugeben, diese Ausgabe schlägt in der gespeicherten Fassung allerdings fehl. Der Grund hierfür wurde im Abschnitt 9*.1 JavaScript Problem* erörtert.

Abbildung 10: Test der Software – CNN

9.3.2 Google

Informationstabelle	Österreichische Startseite der Suchmaschine von Google http://www.google.at/ (PR: 8)
HTML-Fehlerrate	37 Fehler, 2 Warnungen
Reines HTML	Nein
Ajax	Ja
Probleme	-

Abbildung 11: Test der Software – Google

9.3.3 Apple

Informationstabelle	Startseite der Firma Apple http://www.apple.com/at (PR: 7)
HTML-Fehlerrate	4 Fehler, 2 Warnungen
Reines HTML	Nein
Ajax	Nein
Probleme	-

Abbildung 12: Test der Software – Apple

9.3.4 Das Weiße Haus

Informationstabelle	Informationsseite des Präsidenten der USA http://www.whitehouse.gov/ (PR: 9)
HTML-Fehlerrate	54 Fehler, 1 Warnung
Reines HTML	Nein
Ajax	Nein
Probleme	-

Abbildung 13: Test der Software – Das Weiße Haus

9.3.5 YouTube

Informationstabelle	Das größte online Videoportal http://www.youtube.com/ (PR: 9)
HTML-Fehlerrate	114 Fehler, 2 Warnungen
Reines HTML	Nein
Ajax	Ja
Probleme	Die Werbung wird nicht angezeigt, ebenso die unteren Thumbnails der Videos. Diese Bilder werden deswegen nicht angezeigt, weil sie sich auf einem Proxy-Server befinden. Die Originalbilder konnten auch nicht über einen direkten Aufruf erreicht werden.

Abbildung 14: Test der Software – YouTube

9.3.6 Wikipedia

Informationstabelle	Deutschsprachige Startseite von Wikipedia http://de.wikipedia.org/wiki/Wikipedia:Hauptseite (PR: 8)
HTML-Fehlerrate	2 Fehler
Reines HTML	Nein
Ajax	Ja
Probleme	-

Abbildung 15: Test der Software – Wikipedia

9.3.7 Amazon

Informationstabelle	Englischsprachige Startseite des Online-Händlers Amazon http://www.amazon.com/ (PR: 9)
HTML-Fehlerrate	508 Fehler, 117 Warnungen
Reines HTML	Nein
Ajax	Ja
Probleme	-

Abbildung 16: Test der Software – Amazon

9.3.8 JKU

Informationstabelle	Startseite der Johannes Kepler Universität http://www.jku.at/content (PR: 8)
HTML-Fehlerrate	Keine Fehler oder Warnungen
Reines HTML	Nein
Ajax	Ja
Probleme	Die Suchmaske wird nicht korrekt angezeigt, es fehlen die Bilder. Der Grund hierfür wurde im Abschnitt 9*.1 JavaScript-Problem* erörtert.

Abbildung 17: Test der Software – JKU

9.3.9 Newgrounds

Informationstabelle	Eines der größten Flash-Portale im Internet http://www.newgrounds.com/ (PR: 7)
HTML-Fehlerrate	1 Fehler
Reines HTML	Nein
Ajax	Ja
Probleme	Die Werbeflächen auf dieser Webseite konnten nicht gesichert werden. Der Grund hierfür wurde im Abschnitt 9*.1 JavaScript-Problem* erörtert.

Abbildung 18: Test der Software – Newgrounds

9.3.10 T-Mobile

Informationstabelle	Startseite des Telefonanbieters T-Mobile http://www.t-mobile.de/ (PR: 8)
HTML-Fehlerrate	71 Fehler, 30 Warnungen
Reines HTML	Nein
Ajax	Ja
Probleme	-

Abbildung 19: Test der Software – T-Mobile

9.4 Kleine Webseiten

Die hier angeführten Webseiten haben einen Pagerank, der kleiner als 7 ist. Abbildung 20 bis Abbildung 26 beinhalten die Ergebnisse des Evaluierungsprozesses bezüglich kleiner Webseiten.

9.4.1 Microsoft

Informationstabelle	Österreichische Startseite von Microsoft http://www.microsoft.com/de/at (PR: 6)
HTML-Fehlerrate	126 Fehler, 23 Warnungen
Reines HTML	Nein
Ajax	Ja
Probleme	Im unteren Bereich der Webseite fehlt ein Werbeblock, der mittels JavaScript aufgebaut wird. Der Grund hierfür wurde im Abschnitt *9.1 JavaScript Problem* erörtert.

Abbildung 20: Test der Software – Microsoft

9.4.2 Huscarl

Informationstabelle	Startseite des österreichischen Mittelaltermagazins Huscarl http://huscarl.at (PR: 3)
HTML-Fehlerrate	10 Fehler, 12 Warnungen
Reines HTML	Nein
Ajax	Nein
Probleme	-

Abbildung 21: Test der Software – Huscarl

9.4.3 Bluthunde

Informationstabelle	Eine mit Drupal 7 erstellte Webseite http://www.bluthunde.at (PR: 0)
HTML-Fehlerrate	8 Fehler
Reines HTML	Nein
Ajax	Nein
Probleme	-

Abbildung 22: Test der Software – Bluthunde

9.4.4 ORF

Informationstabelle	Startseite des ORF http://orf.at/ (PR: 6)
HTML-Fehlerrate	1 Fehler, 1 Warnung
Reines HTML	Nein
Ajax	Nein
Probleme	-

Abbildung 23: Test der Software – ORF

9.4.5 GomTV

Informationstabelle	Englischsprachige Startseite der Global Starcraft League http://www.gomtv.net/ (PR: 5)
HTML-Fehlerrate	Das Dokument konnte vom W3C-Validator nicht validiert werden, da es Zeichen enthält, die nicht als UTF-8 interpretiert werden können.
Reines HTML	Nein
Ajax	Ja
Probleme	Das Facebook-Plugin auf der linken Seite lädt keinen Inhalt. Der Grund hierfür wurde im Abschnitt 9.*1 JavaScript-Problem* erörtert.

Abbildung 24: Test der Software – GomTV

9.4.6 JavaForum

Informationstabelle	Deutschsprachiges Forum bezüglich Java http://www.java-forum.org (PR: 4)
HTML-Fehlerrate	24 Fehler
Reines HTML	Nein
Ajax	Ja
Probleme	-

Abbildung 25: Test der Software – JavaForum

9.4.7 Stackoverflow

Informationstabelle	Englischsprachiges Forum bezüglich Themen der Informatik http://stackoverflow.com/ (PR: 6)
HTML-Fehlerrate	0 Fehler, 1 Warnung
Reines HTML	Nein
Ajax	Ja
Probleme	Das Begrüßungsfenster, das mittels JavaScript realisiert wurde, kann nicht angezeigt werden. Der Grund hierfür wurde im Abschnitt 9*.1 JavaScript-Problem* erörtert.

Abbildung 26: Test der Software – Stackoverflow

9.5 Evaluierung im Detail

In diesem Unterkapitel werden die Sicherungen von zwei unterschiedlichen Webseiten im Detail analysiert. Während Webstamp beinahe kein Problem hatte, die erste Webseite zu sichern, so wurde die zweite bewusst gewählt, um die aktuellen Schwächen beziehungsweise Einschränkungen zu verdeutlichen.

Die KUSSS-Webseite (http://www.kusss.jku.at) wurde als problemlose und die CNN-Webseite (http://www.cnn.edition.com) als problembehaftete Webseite ausgewählt. Um die einzelnen Problemsituationen zu verdeutlichen, wurden im folgenden Unterkapitel einige Screenshots aufgeführt, die eine Gegenüberstellung der Webseite am Server und der aus der Sicherung ermöglichen.

Weiter wurde mit Tabelle 9 eine Tabelle erstellt, die einen Überblick auf die gesicherten Dateitypen und ihre Anzahl gibt. Danach folgt mit Abbildung 29 eine Liste, die eine eingeschränkte Auflistung aller gesicherten Daten darstellt.

9.5.1 Gesammelte Daten

Während des Sicherungsprozesses wird üblicherweise eine große Anzahl an Dateien generiert. Tabelle 9 gibt einen Überblick darüber, aus welchen Dateitypen sich eine solche Sicherung zusammensetzt. Die Zahlen in dieser Tabelle geben an, wie oft ein bestimmter Dateityp vorkommt, allerdings gibt es dabei zu beachten, dass diese Zahlen nur die forensische Sicherung betreffen. Die Standalone Sicherung würde die vorhandenen Zahlenwerte verdoppeln.

	CNN	KUSSS
Dateien	211	57
Systemdateien	5	5
Grafiken	186	39
JavaScript	3	5
HTML	2	1
CSS	1	7
XML	2	0
RSS	1	0
Unbekannte Dateiendungen	11	0
Erzeugte Ordner	99	9

Tabelle 9: Generierte Sicherungsdateien

9.5.2 KUSSS

Als Beispiel für eine beinah fehlerlose Sicherung wurde die KUSSS-Webseite herangezogen, die unter der URL *http://www.kusss.jku.at* zu erreichen ist.

Um einen Vergleich zwischen Original und Sicherung zu ermöglichen, wurde die Webseite in zwei Bereiche, einen oberen und einen unteren, unterteilt. Während Abbildung 27 den oberen Teil zeigt, so stellt Abbildung 28 den unteren Teil dar. Abschließend wird eine Auflistung aller Dateien gegeben, die sich nach der Sicherung im lokalen Dateisystem befanden. Die einzigen Dateien, die dort aufgrund platzsparender Maßnahmen nicht aufgelistet wurden, sind die Grafiken.

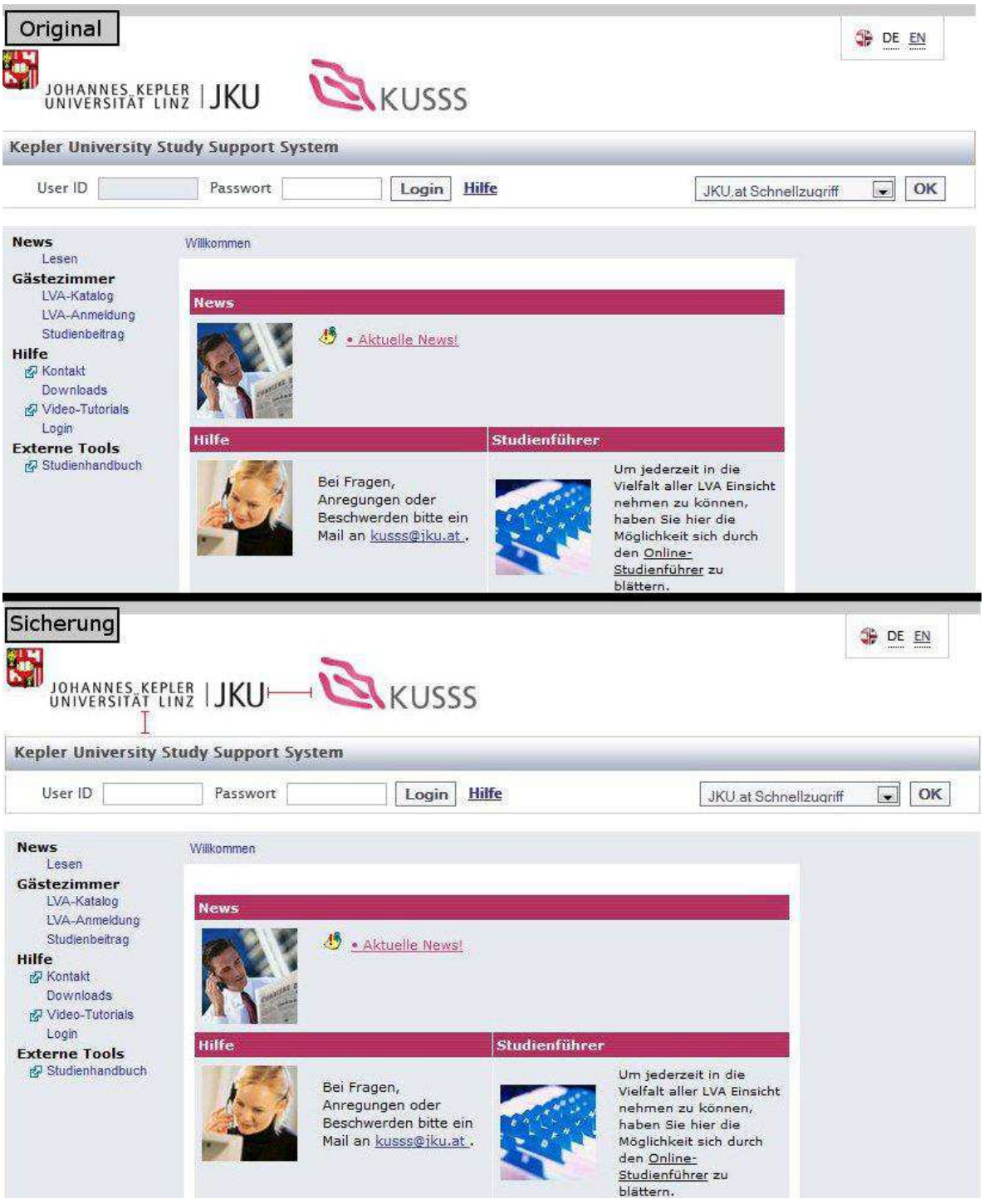

Abbildung 27: Beispiel, KUSSS-Sicherung, Oberer Teil

Abbildung 28: Beispiel, KUSSS-Sicherung, Unterer Teil

C:\DL\KUSSS_certificate.log
C:\DL\KUSSS_checksum.log
C:\DL\KUSSS_default.log
C:\DL\KUSSS_filename.log
C:\DL\KUSSS_header.log
C:\DL\KUSSS\kusss.jku.at\common\kusss\scripts\change.js
C:\DL\KUSSS\kusss.jku.at\common\kusss\scripts\checkall.js
C:\DL\KUSSS\kusss.jku.at\common\kusss\scripts\login.js
C:\DL\KUSSS\kusss.jku.at\common\kusss\scripts\popup.js
C:\DL\KUSSS\kusss.jku.at\common\kusss\scripts\rulemgmt.js
C:\DL\KUSSS\kusss.jku.at\common\kusss\styles\5browser.css
C:\DL\KUSSS\kusss.jku.at\common\kusss\styles\breadCrumbTrail.css
C:\DL\KUSSS\kusss.jku.at\common\kusss\styles\kusss.css
C:\DL\KUSSS\kusss.jku.at\common\kusss\styles\kusss_interim.css
C:\DL\KUSSS\kusss.jku.at\common\kusss\styles\login.css
C:\DL\KUSSS\kusss.jku.at\common\kusss\styles\menu.css
C:\DL\KUSSS\kusss.jku.at\common\kusss\styles\tabbedPanel.css
C:\DL\KUSSS\kusss.jku.at\kusss\index.action.file
C:\DL\KUSSS_live\kusss.jku.at\common\kusss\scripts\change.js
C:\DL\KUSSS_live\kusss.jku.at\common\kusss\scripts\checkall.js
C:\DL\KUSSS_live\kusss.jku.at\common\kusss\scripts\login.js
C:\DL\KUSSS_live\kusss.jku.at\common\kusss\scripts\popup.js
C:\DL\KUSSS_live\kusss.jku.at\common\kusss\scripts\rulemgmt.js
C:\DL\KUSSS_live\kusss.jku.at\common\kusss\styles\5browser.css
C:\DL\KUSSS_live\kusss.jku.at\common\kusss\styles\breadCrumbTrail.css
C:\DL\KUSSS_live\kusss.jku.at\common\kusss\styles\kusss.css
C:\DL\KUSSS_live\kusss.jku.at\common\kusss\styles\kusss_interim.css
C:\DL\KUSSS_live\kusss.jku.at\common\kusss\styles\login.css
C:\DL\KUSSS_live\kusss.jku.at\common\kusss\styles\menu.css
C:\DL\KUSSS_live\kusss.jku.at\common\kusss\styles\tabbedPanel.css
C:\DL\KUSSS_live\kusss.jku.at\kusss\index.action.file

Abbildung 29: Beispiel, KUSSS-Sicherung, Dateien

9.5.2.1 Analyse

Die Sicherung unterscheidet sich in zwei Punkten vom Original.

Der erste Unterschied wurde in Abbildung 27 mit roten Abstandslinien markiert. Im Unterschied zur Originalversion ist der Abstand zwischen der JKU-Schrift und dem KUSSS-Logo größer. Ebenso der Abstand zwischen der Schrift im oberen Teil der Homepage und dem Beginn des Abschnittes "Kepler University Study Supporting System". Diese minimale Abweichung vom Layout des Originals tritt gelegentlich immer wieder auf und ist dem verwendeten HTML-Parser Jsoup zuzuschreiben. Das Problem liegt an der Verwendung von unterschiedlichen Zeichensätzen. Auf die forensische Sicherung wirkt sich dieses Problem nicht aus.

Der zweite Unterschied wird in Abbildung 28 dargestellt. Die Anzahl der Benutzer, die online sind, unterscheidet sich hierbei. Auf den ersten Blick sieht es so aus, als ob es selbstverständlich und auch kein Problem sei, wenn sich dieser Wert unterscheidet. Allerdings könnte es sich hierbei ebenso um wichtige Daten handeln wie zum Beispiel einen neuen Eintrag. Dadurch soll verdeutlicht werden, dass die Sicherung immer dem Stand entspricht, zu dem sie ausgeführt wurde, und nicht dem Zeitpunkt, zu dem man sich entschied, eine Sicherung durchzuführen.

9.5.3 CNN

Um einen Überblick über die Einschränkungen von Webstamp zu geben, wurde die CNN-Webseite vom 25. April 2012 herangezogen. Um die Analyse der Sicherung zu ermöglichen, wurde diese Webseite in vier Teile zerlegt, welche die nicht korrekt dargestellten Inhalte zeigen.

Es ist bei der folgenden Evaluierung wichtig, dass die hier auftretenden Fehler nur die Standalone-Version der Sicherung betreffen, die darauf ausgelegt wurde, die gesicherten Inhalte möglichst originalgetreu wiederzugeben. Das Fehlen eines Inhaltes bedeutet somit

nicht sofort, dass der Inhalt nicht gesichert wurde, es bedeutet meist nur, dass ein Anzeigeproblem vorliegt, das auf ein JavaScript zurückzuführen ist.

9.5.3.1 Analyse: Overlay

Abbildung 30: Beispiel, CNN-Sicherung, Overlay

Am obersten Bereich der Webseite wird nach dem Laden des Inhaltes ein etwa ein Zentimeter großes Overlay mittels JavaScript angezeigt. Das Overlay wird dazu benutzt, den User zu fragen, ob die aktuell betrachtete internationale Version der Homepage standardmäßig angezeigt werden soll. In der gesicherten Version wird dieses Overlay nicht angezeigt.

Beim Betrachten der gesicherten Daten fällt auf, dass die schwarze Hintergrundgrafik gesichert wurde, ebenso konnten die verwendeten Textstücke in dem JavaScript *globallib.intl-min.js* gefunden werden. Der Block beziehungsweise das *div*-Element, in dem sich das Overlay befindet, können aus Codestück 11 entnommen werden. Daraus wird ersichtlich, dass der Block nicht angezeigt wird, da das Attribut *display* auf *none* gesetzt wurde.

```
<div id="cnn_hdr-prompt" style="display:none;">
  <div class="hdr-wrap" id="cnn_hdr-promptcntnt"></div>
</div>
```

Codestück 11: CNN Overlay

Selbst wenn jedoch dieser Block angezeigt werden würde, so würde er keinen Text enthalten. Bei der genauen Untersuchung der Original-Webseite wurde festgestellt, dass hier mehrere Blöcke fehlen, die durch ein JavaScript angezeigt werden, nachdem der Inhalt im Browser geladen wurde. Das entsprechende JavaScript liegt in einer unformatierten Form vor und besteht aus 30 Zeilen zu je ca. 4.000 Zeichen. Aufgrund dieser Größe ist es schwierig, die genaue Stelle zu finden, die für dieses Problem verantwortlich ist, ohne das entsprechende Skript zuvor aufzubereiten.

Beim Betrachten des Quellcodes fielen sofort zwei bekannte Probleme ins Auge. Es werden sehr viele String-Operationen mit oft unvollständigen oder ungültigen URLs durchgeführt. Webstamp kann mit unvollständigen URLs nichts anfangen, genauso wenig wie mit ungültigen URLs.

Der Grund, warum einige ungültige URLs in diesem Skript vorkommen, ist, das hier eine eigene Notation verwendet wird, in der mehrere URLs hintereinander aufgeführt werden. Bei der Ausführung im Browser werden diese URLs dann in ihre Bestandteile zerlegt. Codestück 12 zeigt eine solche URL. Der Grund, warum das Overlay keinen Text enthält, hängt also damit zusammen, dass unvollständige URLs nicht gesichert werden können.

```
http:/cnn/tmpl_asset/static/intl_global/311/js/globallib.intl-
min.jsz.cdn.turner.com/cnn/tmpl_asset/static/intl_global/311/js/globallib.intl-
min.js//z.cdn.turner.com/cnn/tmpl_asset/static/intl_global/311/js/https
```

Codestück 12: CNN zusammengesetzte URL

9.5.3.2 *Analyse: Editor‘s Choice*

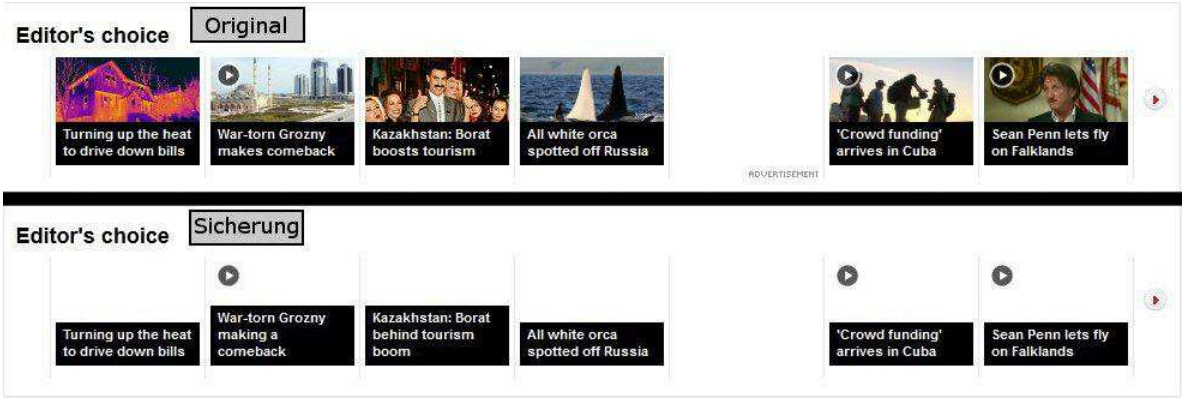

Abbildung 31: Beispiel, CNN-Sicherung, Ausgewählte Artikel

Auf diesem Bereich der Homepage werden besondere Artikel mit einer Grafik und einem kurzem Text hervorgehoben. Abbildung 31 zeigt, dass die Bilder in der Standalone-Version der Sicherung nicht angezeigt werden. Interessanterweise wurden die anscheinend fehlenden Bilder aber dennoch gesichert. Das Problem liegt an einer Style-Anweisung, welche die Bilder mit dem Attribut *display: none* versteckt. Das entsprechende Codestück 13 zeigt diese Situation, die bei allen Bildern in diesem Bereich auftritt.

```
<div class="cnn_fabcatimg">
        <a href="http://edition.cnn.com/2012/04/23/opinion/france-election/index.html?hpt=hp_mid">
                <img src="file:///C:/DL/CNN_live/i.cdn.turner.com/cnn/.e/img/3.0/1px.gif" alt="" border="0"
                class="cnnContentImg" height="68" style="background:transparent
                url(file:///C:/DL/CNN_live/i2.cdn.turner.com/cnn/dam/assets/120423095438-france-papers-video-tease.jpg)
                0 0 no-repeat;display:none" width="120"/>
        </a>
</div
```

Codestück 13: CNN Editor’s Choice

Bei der weiteren Untersuchung der gesicherten JavaScripte und der Stylesheets wurde gezielt nach dem *cnn_fabcatimg* Tag gesucht. In keiner der gesicherten Dateien trat dieser Tag auf. Auch der weiter umschließende Tag konnte nicht gefunden werden. Dies lässt darauf schließen, dass der entsprechende Wert serverseitig, bereits bei der Auslieferung, gesetzt wurde.

9.5.3.3 Analyse: Werbeblock

Wie auf allen großen Webseiten, so wird auch auf der CNN-Webseite Werbung geschaltet. Als Platz hierfür dient die rechte Seite, wo sich zusätzlich zur Werbung auch ein Facebook-Plugin befindet. Abbildung 32 zeigt den gravierenden Unterschied zwischen dem Original und der Sicherung.

Das Facebook-Plugin wird mitsamt allen Style-Informationen korrekt angezeigt, bei dem Twitter-Plugin fehlen die entsprechenden Style-Informationen aber. Der Grund dafür findet sich in einem JavaScript auf der Index-Seite der Sicherung wieder. Hier wird wie aus Codestück 14 ersichtlich der Verweis auf das JavaScript zur Steuerung des Twitter-Buttons nicht vollständig angegeben. Da der Link zu diesem JavaScript aber mehr als einmal vorkommt und in mindestens einem dieser Fälle vollständig war, wurde er gesichert. Würde man den relativen Pfad nun absolut machen, so käme es zur korrekten Darstellung. Das Problem beruht also darauf, dass unvollständige URLs nicht gesichert und auch nicht absolut gemacht werden können, wenn sie innerhalb eines JavaScripts vorkommen. Relative URLs die außerhalb von JavaScript vorkommen, können vom Webstamp hingegen problemlos absolut gemacht werden.

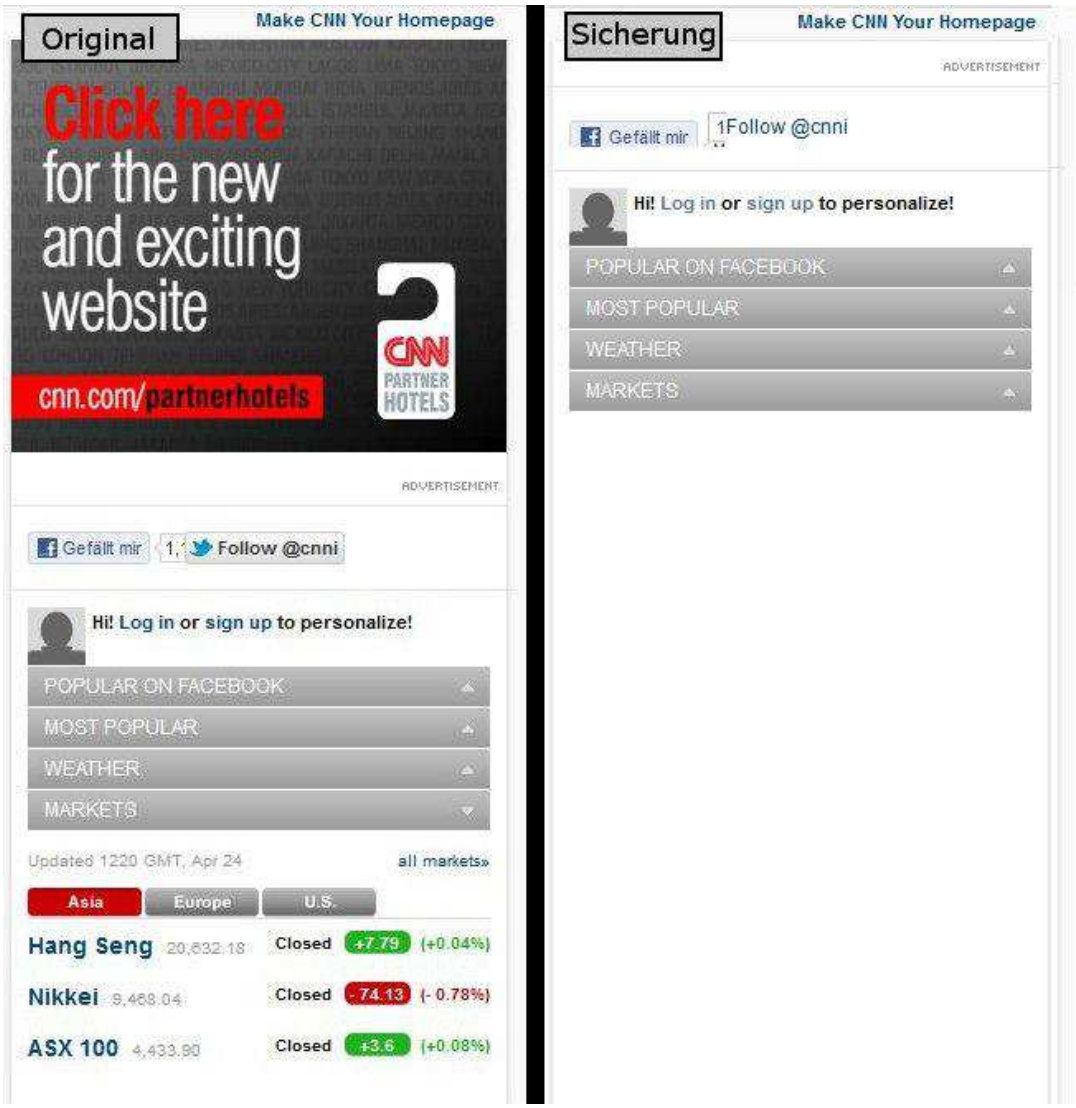

Abbildung 32: Beispiel, CNN-Sicherung, Werbung

```
<script>
 !function(d,s,id){
            var js,fjs=d.getElementsByTagName(s)[0];
            if(!d.getElementById(id)){
                        js=d.createElement(s);
                        js.id=id;
                        js.src="//platform.twitter.com/widgets.js";
                        fjs.parentNode.insertBefore(js,fjs);
            }
            }(document,"script","twitter-wjs");
</script>
```

Codestück 14: CNN Werbung, Twitter

Die Werbung, die sich oberhalb des Twitter- und des Facebook-Plugins befindet, wird nicht angezeigt. Normalerweise sollte sich hier ein IFrame mit der entsprechenden Werbung befinden. Welche Werbung sich darin befindet, wird von einem JavaScript auf der Index-Seite der Sicherung festgelegt. Der Grund, warum keine Werbebilder angezeigt und auch keine gesichert wurden, liegt daran, dass die entsprechenden URLs stückchenweise zusammengesetzt wurden.

Nach dem Webeblock kommt ein personalisierter Bereich. Während der Login-Bereich hier korrekt dargestellt wird, gibt es mit den sich darunter befindlichen Menüpunkten ein Problem. Standardmäßig sollte hier der Menüpunkt *MARKETS* aufgeklappt sein, bei der Sicherung hingegen sind alle Menüpunkte zugeklappt. Der Grund liegt im verwendeten JavaScript, welches, nachdem die Seite im Browser angezeigt wurde, den entsprechenden Menüpunkt ausklappen sollte. Das ausgeklappte Menü wird nicht angezeigt, da innerhalb des JavaScripts mit unvollständigen URLs gearbeitet wird welche die korrekte Ausführung des JavaScripts innerhalb der Sicherung verhindern. Dies führt dazu dass sie Sichtbarkeit des entsprechenden Blocks auf *display: none* gesetzt wurde. Würde man diese Eigenschaft händisch auf beispielsweise *display: block* umändern, so käme es zur korrekten Darstellung. Die nicht angezeigten Elemente wurden somit gespeichert, lediglich die Anzeige ist aufgrund des JavaScripts fehlerhaft.

9.5.3.4 Analyse: Footer

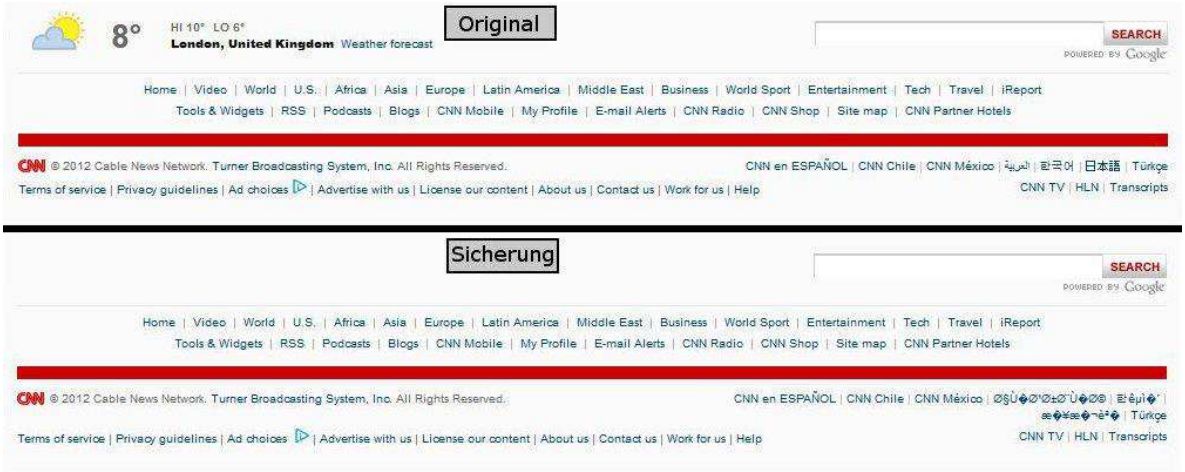

Abbildung 33: Beispiel, CNN-Sicherung, Footer

Im Footer der CNN-Webseite befindet sich ein eigener Block, der Informationen über das gegenwärtige Wetter liefert.

Das den Wetterblock umschließende DIV hat die Eigenschaft *visibility=hidden*. Dies führt dazu, dass der entsprechende Block nicht angezeigt wird. Durch das Sichtbarmachen des Blocks wird eine Meldung angezeigt (Loading weather data …), die nahelegt dass es sich hier um eine AJAX-Anwendung handelt, welche die Wetter-Daten nach dem Empfang der Webseite nachlädt. Da AJAX-Anwendungen nicht mit Webstamp kompatibel sind, kann dieser Block nicht angezeigt werden.

9.5.3.5 Gesicherte Dateien

Die nun folgenden Abbildung 34 und Abbildung 35 stellen eine Auflistung aller gesicherter Daten dar, exklusive der Bilddateien.

```
C:\DL\CNN_certificate.log
C:\DL\CNN_checksum.log
C:\DL\CNN_default.log
C:\DL\CNN_filename.log
C:\DL\CNN_header.log
C:\DL\CNN\edition.cnn.com\CNN International Home Page.file
C:\DL\CNN\edition.cnn.com\cnnintl_adspaces\3.0\homepage\main\bot1.120x90.ad
C:\DL\CNN\edition.cnn.com\cnnintl_adspaces\3.0\homepage\main\bot2.120x90.ad
C:\DL\CNN\edition.cnn.com\cnnintl_adspaces\3.0\homepage\main\bot3.120x90.ad
C:\DL\CNN\edition.cnn.com\cnnintl_adspaces\3.0\homepage\main\bot4.120x90.ad
C:\DL\CNN\edition.cnn.com\cnnintl_adspaces\3.0\homepage\main\bot5.120x90.ad
C:\DL\CNN\edition.cnn.com\cnnintl_adspaces\3.0\homepage\main\bot6.120x90.ad
C:\DL\CNN\edition.cnn.com\cnnintl_adspaces\3.0\homepage\main\bot7.120x90.ad
C:\DL\CNN\edition.cnn.com\cnnintl_adspaces\3.0\homepage\main\bot8.120x90.ad
C:\DL\CNN\edition.cnn.com\cnnintl_adspaces\3.0\homepage\main\bot9.120x90.ad
C:\DL\CNN\edition.cnn.com\index.html
C:\DL\CNN\edition.cnn.com\tools\search\cnncom.xml
C:\DL\CNN\edition.cnn.com\tools\search\cnncomvideo.xml
C:\DL\CNN\facebook.com\plugins\like.php_href=http3A2F2Ffacebook.com2Fcnninternational&send=false&layout=button_count
&width=450&show_faces=false&action=like&colorscheme=light&font=arial&height
C:\DL\CNN\i.cdn.turner.com\cnn\.element\css\3.0\png_fix.htc
C:\DL\CNN\platform.twitter.com\widgets.js
C:\DL\CNN\rss.cnn.com\rss\edition.rss
C:\DL\CNN\z.cdn.turner.com\cnn\tmpl_asset\static\intl_global\311\js\globallib.intl-min.js
C:\DL\CNN\z.cdn.turner.com\cnn\tmpl_asset\static\intl_homepage\709\css\intlhplib-min.css
C:\DL\CNN\z.cdn.turner.com\cnn\tmpl_asset\static\intl_homepage\709\js\intlhplib-min.js
```

Abbildung 34: Beispiel, CNN-Sicherung, Systemdateien + forensische Sicherung

Während in Abbildung 34 die Systemdateien inklusive der Dateien für die forensische Sicherung aufgelistet sind, so werden in Abbildung 35 die Dateien der Standalone-Sicherung aufgelistet.

C:\DL\CNN_live\edition.cnn.com\CNN International Home Page.file
C:\DL\CNN_live\edition.cnn.com\cnnintl_adspaces\3.0\homepage\main\bot1.120x90.ad
C:\DL\CNN_live\edition.cnn.com\cnnintl_adspaces\3.0\homepage\main\bot2.120x90.ad
C:\DL\CNN_live\edition.cnn.com\cnnintl_adspaces\3.0\homepage\main\bot3.120x90.ad
C:\DL\CNN_live\edition.cnn.com\cnnintl_adspaces\3.0\homepage\main\bot4.120x90.ad
C:\DL\CNN_live\edition.cnn.com\cnnintl_adspaces\3.0\homepage\main\bot5.120x90.ad
C:\DL\CNN_live\edition.cnn.com\cnnintl_adspaces\3.0\homepage\main\bot6.120x90.ad
C:\DL\CNN_live\edition.cnn.com\cnnintl_adspaces\3.0\homepage\main\bot7.120x90.ad
C:\DL\CNN_live\edition.cnn.com\cnnintl_adspaces\3.0\homepage\main\bot8.120x90.ad
C:\DL\CNN_live\edition.cnn.com\cnnintl_adspaces\3.0\homepage\main\bot9.120x90.ad
C:\DL\CNN_live\edition.cnn.com\index.html
C:\DL\CNN_live\edition.cnn.com\tools\search\cnncom.xml
C:\DL\CNN_live\edition.cnn.com\tools\search\cnncomvideo.xml
C:\DL\CNN_live\facebook.com\plugins\like.php_href=http3A2F2Ffacebook.com2Fcnninternational&send=false&layout=button_count&width=450&show_faces=false&action=like&colorscheme=light&font=arial&height
C:\DL\CNN_live\i.cdn.turner.com\cnn\.element\css\3.0\png_fix.htc
C:\DL\CNN_live\platform.twitter.com\widgets.js
C:\DL\CNN_live\rss.cnn.com\rss\edition.rss
C:\DL\CNN_live\z.cdn.turner.com\cnn\tmpl_asset\static\intl_global\311\js\globallib.intl-min.js
C:\DL\CNN_live\z.cdn.turner.com\cnn\tmpl_asset\static\intl_homepage\709\css\intlhplib-min.css
C:\DL\CNN_live\z.cdn.turner.com\cnn\tmpl_asset\static\intl_homepage\709\js\intlhplib-min.js

Abbildung 35: Beispiel, CNN-Sicherung, Standalone-Sicherung

Wie aus den beiden Abbildungen hervorgeht werden bei beiden Sicherungen dieselben Dateien gespeichert. Der Unterschied liegt lediglich darin, dass die Standalone Sicherung im Gegensatz zur forensischen Sicherung keine Log-Dateien enthält.

10 Sicherheitsüberlegungen

Dieses Kapitel behandelt die generellen Sicherheitsüberlegungen im Hinblick auf die Applikation.

10.1 Wie können sich Webseiten wehren

Im Rahmen der ersten Präsentation dieser Arbeit stellte sich die Frage, wie sicher und vertrauenswürdig die Methode des Anhängens von Parametern ist. Weiter wurde die Befürchtung geäußert, dass Webserver die Parameter aller abgerufenen Dateien zählen und bei einer Abweichung negativ reagieren könnten.

Bei den durchgeführten Tests wurde eine Vielzahl an Webseiten getestet, allerdings konnte kein einziger Server gefunden werden, der negativ auf diese Methode reagierte. Ein Auszug der getesteten Webseiten findet sich im *Kapitel 9 Evaluierung* wieder.

Neben der Durchführung dieser Evaluierung versuchte ich Methoden zu finden, um Webstamp auszuhebeln, allerdings war die einzig vielversprechende das Führen einer Datenbank, die zu jeder vorhandenen Ressource speicherte, wo sie mit welchen Parametern verwendet wurde.

Bei näherer Betrachtung dieses Ansatzes treten sehr schnell einige Probleme auf. Kommt beispielsweise ein Bild auf einer Webseite öfter als einmal vor, muss dieses Bild in jeder vorhandenen Konfiguration in der Datenbank vorliegen. Wird nun ein CMS-System eingesetzt und ein Autor fügt ein Bild in die entsprechende Webseite ein, muss dieses auch in die Datenbank eingetragen werden. Hierbei wird schnell klar, dass der administrative Aufwand, der hinter so einer Lösung steckt, enorm ist.

Statische Webseiten könnten sich so mit extrem hohem Aufwand gegen diese Methode wehren, bei dynamischen Webseiten sieht dies allerdings ganz anders aus. Falls ein Webserver dynamisch verschiedene Parameter an eine Ressource anhängt, müsste der Inhalt der entsprechenden Datenbank auch dynamisch aktualisiert werden.

Zusammenfassend kann gesagt werden, dass eine theoretische Lösung existiert, um die Methode des Anhängens von Parametern zu umgehen. Hierbei wird jede Anfrage, falls sie um unbekannte Elemente erweitert wurde, abgelehnt. Dies führt zu einer Fehlerseite. In der Praxis trat dieser Verteidigungsmechanismus allerdings nicht auf.

10.2 Zeitstempeldienste

Webstamp generiert bei jedem Sicherungsvorgang eine Menge an Daten, die entweder zur gesicherten Webseite zählen (HTML, Ressourcen, Stylesheets) oder dafür verwendet werden, um den Vorgang zu einem beliebigen Zeitpunkt nachvollziehen zu können. All diese Daten werden lokal gespeichert, und so stellt sich die Frage, wie es möglich ist, sicherzustellen, dass diese Daten nach der Sicherung nicht verändert werden können. Die Lösung zu diesem Problem liefert das Konzept der Zeitstempeldienste.

„Ein Zeitstempeldienst versieht ein elektronisches Dokument, das ihm vorgelegt wird, mit dem aktuellen Datum und der aktuellen Uhrzeit und signiert diese Daten. […] Zeitstempeldienste lassen sich aber auch für die Nichtabstreitbarkeit von elektronisch geschlossenen Verträgen und Notariatsdiensten sowie für den Nachweis der zeitlichen Korrektheit und damit der Urheberschrift von digitalen Dokumenten wie Patentschriften verwenden.“ [10]

Bei dem Dokument, das an den Zeitstempeldienst übermittelt wird, handelt es sich um den Hashwert einer einfachen Textdatei, die den Speicherort, den Dateinamen sowie den Hash-Wert aller gesicherten Dateien enthält. Die Daten, die zur späteren Nachvollziehbarkeit erzeugt werden, befinden sich ebenfalls in dieser Liste. Die Datei, die all diese Informationen enthält, befindet sich im Root-Verzeichnis der Sicherung und trägt den Dateinamen *(Projektname)_certificate.log*. Bei einem Projekt, das den Namen *Google* trägt würde dies den Namen *Google_checksum.log* erzeugen.

Der Zeitstempel liefert nach der Übermittlung des Hashwertes ein Zertifikat zurück, das im ASN.1-Format vorliegt. Dieses Zertifikat wird in kodierter Form im Root-Verzeichnis des Projektes abgelegt und trägt den Dateinamen *(Projektname)_certificate.log*. Bei dem

Projekt *Google* führt dies zu dem Dateinamen *Google_certificate.log*. Ein im ASN.1-Format vorliegendes Zertifikat wird in Codestück 15 dargestellt. Hier handelt es sich um das Zertifikat, das nach dem Sicherungsvorgang von www.google.at erhalten wurde. Die eigentliche Signatur der Hash-Liste wurde hierbei hervorgehoben.

```
[[[[0]2, 306159427, [1.2.840.113549.1.1.5, NULL], [

[[2.5.4.6, DE]], [[2.5.4.10, DFN-Verein]], [[2.5.4.11, DFN-PKI]], [[2.5.4.3, DFN-Verein CA Services]]

], [110715111707Z, 160713111707Z], [[[2.5.4.6, DE]], [[2.5.4.10, DFN-Verein]], [[2.5.4.11, DFN-PKI]], [[2.5.4.3, PN: Zeitstempel]]],[[1.2.840.113549.1.1.1,NULL],
#0382010F003082010A0282010100A9AC33B296DA7177999D464F47AA4A40D57D58DCFD93BEAE68913AB75CB77FE36C4
B52B3B55A53CCE10F70880A81ABA4FFDC1D4826FE645CBABCD1E0B4ECEFF702F6FB378670128EADBE39A4A9E484C1
D01F95FCFCBD44CA091DCC344E0356CA8967F54F7F6ACC0DD5AF8C1A4F77003FE01C3B98D6611D52B3FE432962544E
142CC6F99163CCB7798BB8D4AEA948D0CD6F72B740915B87CA2824AC9EC958AB0E5EACB36A7A66BE091E826F862849
026AA911E3B1A84487F6654AAD7F3BE4D1D9D312B2F9FCD7C69836AE893060393A47B310A6A4B03EEEA6C8659DF577
82FA75855007D5FFB622FF8D229EDD57C0771149B7FC827780FCDE0C02F82BC2977D250203010001],

[3][[2.5.29.19, #3000], [2.5.29.15, #030206c0], [2.5.29.37, TRUE, #300a06082b06010505070308], [2.5.29.14, #0414ec88dc78663109b4892d7ddb1b9cacd80642aed9], [2.5.29.35, #301680141da9f18626764dcf5dfd50a36eebf1bc22756deb], [2.5.29.31,
#3081863041a03fa03d863b687474703a2f2f636470312e7063612e64666e2e64652f676c6f62616c2d73657276696365732d63612f7075622f63726c2f636163726c2e63726c3041a03fa03d863b687474703a2f2f636470322e7063612e64666e2e64652f676c6f62616c2d73657276696365732d63612f7075622f63726c2f636163726c2e63726c],

[1.3.6.1.5.5.7.1.1,
#3081cf303306082b06010505073001862768747470 3a2f2f6f6373702e7063612e64666e2e64652f4f4353502d5365727665722f4f435350304b06082b06010505073002863f687474703a2f2f636470312e7063612e64666e2e64652f676c6f62616c2d73657276696365732d63612f7075622f6361636572742f6361636572742e637274304b06082b06010505073002863f687474703a2f2f636470322e7063612e64666e2e64652f676c6f62616c2d73657276696365732d63612f7075622f6361636572742f6361636572742e637274]]],

[1.2.840.113549.1.1.5,NULL],
#038201010015EF8C64BE9F746166745CE0FFB655139CD47A298D433DA445781DBBF960DED26A0C0FEE0B6C009F613135
C1A8C8CB60E5E9246B4C0FB695E1615DD2F63E0C61AE13B0F8590955BB0A2C4401536AF5F8A400876007A9F475FD68B8
B2DC48E82E981FD6D0A976CD7E53537611144BEDCEBC53B6B8FF92C723FD3E7F091DAAA63A9BD78AAAC035EB36A0
D73A8A02B41255FF7C4CEB369C97127D3AE4163803E088158A6A53851E351C7C45BF920DCE6F9BA460BCD6E386A9589
05DD12B6BEB7F0F139885302575546E3CD96BDCF213D3449E2B9E534307B42C4B4E19507F3ACCB7A61EA87946321BBF0
F0C0C54F452D36D9C36EDAC6DEB9DB0F34476E3CE43578E]]
```

Codestück 15: ASN.1-kodiertes Zertifikat

Das so erhaltene Zertifikat wird nach dem Erhalt sofort geprüft, um sicherzugehen, dass der Vorgang erfolgreich war. Natürlich kann dieses Zertifikat mithilfe der entsprechenden Funktion in der Software zu einem späteren Zeitpunkt beliebig oft validiert werden.

Für die technische Umsetzung dieses Verfahrens wird die öffentlich frei verfügbare BouncyCastle-Bibliothek verwendet, die unter der URL http://www.bouncycastle.org/ zu erreichen ist. Diese Software bietet Implementierungen für das Kodieren und Dekodieren von ASN.1-Objekten an. Weiter werden Funktionen zur Übermittlung und Validierung von ASN.1-Objekten zur Verfügung gestellt.

10.2.1 Implementierung

Codestück 16 zeigt die Implementierung einer Anfrage an einen Zeitstempeldienst. Hierbei wird von der Liste, die alle gesicherten Dateien inklusive ihren Hash-Werten enthält, selbst ein Hash-Wert erzeugt und an einen Zeitstempeldienst übermittelt. Wichtig ist hierbei, dass der *Content-Type* der Anfrage auf *application/timestamp-query* gesetzt wird. Abschließend kann dieser Zeitstempel durch die Validierungsfunktion der BouncyCastle-Klasse überprüft werden.

```
TimeStampRequest req = reqgen.generate(TSPAlgorithms.SHA1, digest);
byte request[] = req.getEncoded();

URL url = new URL(timeStampServer);
HttpURLConnection con = (HttpURLConnection) url.openConnection();

con.setDoOutput(true);
con.setDoInput(true);
con.setRequestMethod("POST");
con.setRequestProperty("Content-type", "application/timestamp-query");
con.setRequestProperty("Content-length", String.valueOf(request.length));

out = con.getOutputStream();
out.write(request);
out.flush();

TimeStampResponse  response = new TimeStampResponse(con.getInputStream());


response.validate(req);
```

Codestück 16: Timestamp-Implementierung

11 Zusammenfassung

Zu Beginn dieser Arbeit wurde mir die Aufgabe gestellt, eine Software zur Speicherung von einzelnen Webseiten zu entwerfen. Die Schwierigkeit bestand darin, dass die Sicherung forensisch verwertbar sein musste und nur die aktuelle Version gespeichert werden sollte. Bei der Analyse der vorhandenen Systeme wurde ersichtlich, dass zwar bereits einige existierten, diese aber keinen Wert für die Forensik hatten und auch keinen Wert auf Aktualität legten. Für meine Arbeit war es aber wichtig, dass die Sicherung nachvollziehbar und validierbar sein würde.

Als nächstes musste ich mich auf eine Technologie festlegen, in der ich die Software Webstamp entwickeln wollte. Ich entschied mich für Java da ein Browser-Plugin zu einer browserspezifischen Sicherung führen würde und ich dies, aus forensischen Gründern, verhindern wollte. Mit Java war die Möglichkeit gegeben vollständig browserunabhängig zu arbeiten und gleichzeitig eine Software zu entwickeln welche auf verschiedenen Systemen laufen würde.

Bei der Entwicklung der Software nutzte ich die Entwicklungsmethode des Prototypings und fügte nach und nach die geforderten Funktionen hinzu. Bei den ersten Versionen handelte es sich um einen reinen Webcrawler, der Webseiten speicherte. Erst später kamen die forensischen Funktionen hinzu.

Für meine Arbeit war es wichtig alle URLs aus einer gefundenen Webseite zu extrahieren und genau dort lag eines der größten Probleme, da die vorliegenden Webseiten, in fast allen Fällen, syntaktische Fehler enthielten. Dies bedeutete, dass ich einige reguläre Ausdrucke erstellen musste um URLs aus HTML-, CSS- und JavaScript-Dateien zu erhalten. Diese regulären Ausdrucke wurden während des ganzen Entwicklungsprozesses immer wieder überarbeitet und optimiert.

Ein weiteres Problem, dass mich beschäftigte war, dass sich einige URLs nicht 1:1 auf das lokale Dateisystem abbilden ließen. Dies lag einerseits daran, dass die URLs länger als die

erlaubte Dateilänge, in den gängigen Dateisystemen, waren und andererseits daran, dass URLs Zeichen enthalten können, welche nicht in Dateinamen vorkommen dürfen. Um diese Probleme zu lösen entschied ich mich für eine Maximallänge und entsprechende Ersetzungszeichen.

Bei der Speicherung der Webseiten war es erforderlich immer die aktuelle Version zu sichern. Dies bedeutet, dass sichergestellt werden musste, dass der abgerufene Inhalt der Seite auch tatsächlich vorhanden ist und nicht etwa durch einen Proxy dargestellt wurde. Um dieses Ziel zu erreichen entschied ich mich für die Verwendung von Caching-Direktiven und entwickelte die Parameter-Methode. Mithilfe dieser Methode wurde an jede URL ein, dem Server, unbekannter Parameter angehängt um die Caching-Mechanismen von Proxys zu umgehen.

Nachdem die Software alle geforderten Funktionalitäten enthielt begann ich mit einer Evaluierung. Hierfür wurden verschiedene Webseiten ausgewählt, gesichert und die Sicherung analysiert. Bei der Analyse konnte ein wesentliches Problem identifiziert werden; kommen in einem JavaScript relative URLs vor, so können diese nicht gesichert werden. Diese Einschränkung blieb auch in der finalen Version von Webstamp bestehen und lässt Raum für Erweiterungen.

Als die finale Version von Webstamp vorlag überlegte ich mir wie man diese Software noch weiter verbessern könnte. Ein interessantes Feature wäre das Speichern von Benutzersitzungen. Hierfür müssten Wege gefunden werden um mit Cookies, Sessions sowie POST-Daten umgehen zu können. Dadurch wäre es möglich Webseiten zu speichern die das Ergebnis einer Benutzersitzung sind. Meine Gedanken diesbezüglich finden sich im nachfolgendem Kapitel.

11.1 Mögliche Erweiterungen: Speichern von Benutzersitzungen

In diesem Kapitel werden potenzielle Erweiterungen der Software Webstamp erörtert. Es handelt sich dabei um Funktionalitäten, die in der vorliegenden Version nicht enthalten sind.

Der Input, den Webstamp zum Speichern von Webseiten erhält, besteht aus einer einfachen URL ohne zusätzliche Informationen. Dies macht es unmöglich, eine Webseite zu speichern, die das Ergebnis einer Benutzersitzung ist, da es keine Informationen über den vorhergehenden Zustand gibt. Es folgt nun eine Auflistung der einzelnen Szenarien von Benutzersitzungen sowie Lösungsvorschläge, wie diese mit Webstamp durch eine Erweiterung der Software gesichert werden könnten.

11.1.1 Cookies

Cookies sind kleine Dateien, die beim Besuch einer Webseite auf dem lokalen Rechner abgelegt werden können. Meist beinhalten diese Dateien Informationen aus abgeschickten Formularen oder werden dazu verwendet, Webseiten zu personalisieren. Der für diese Arbeit wesentliche Aspekt ist dass Cookies dafür verwendet werden können, um eine Benutzersitzung zu realisieren. Dies bedeutet, dass der Aufruf der Webseite ein unterschiedliches Ergebnis liefert, je nachdem, ob bei der Anfrage an den Webserver das Cookie übermittelt wurde oder auch nicht.

Um Webseiten speichern zu können, welche Cookies verwenden, müsste beim ersten Aufruf der Webseite das entsprechende Cookie lokal angelegt und bei jedem einzelnen HTTP-Request ebenfalls übermittelt werden. Falls die entsprechende Webseite mehrere Cookies verwendet, muss dies für jedes einzelne erfolgen. Die Implementierung dieser Funktionalität wäre durchaus möglich, da der HTML-Parser (Jsoup), der bereits innerhalb von Webstamp verwendet wird, Funktionen zur Verwaltung von Cookies zur Verfügung stellt.

Webstamp nimmt zurzeit keine Cookies an und verwendet sie auch nicht im Rahmen des Sicherungsprozesses. Ein Problem, das sich durch die Einbeziehung von Cookies ergeben würde, ist ihre Übertragung vom Browser zu Webstamp. Die Zeiten, in denen Cookies als Textdateien lokal abgelegt wurden, sind mit den aktuellen Browsergenerationen vorbei, nun werden sie in eigenen Datenbanken abgelegt. Insofern müssten die Cookies aus diesem Datenbestand extrahiert und an Webstamp übermittelt werden. Webstamp müsste diese

Daten dann in ein leserliches Format bringen und bei der Anfrage an den Server mitschicken.

11.1.2 Formulare (POST-Daten)

Eine weitere Möglichkeit, um eine Benutzersitzung zu realisieren, ist die Verwendung von POST-Daten. Dies sind im Grunde Variablen, die beim Wechsel von einer Webseite zur nächsten übertragen werden. Hier könnte es sich beispielsweise um die Daten eines ausgefüllten Formulars handeln, die auf der nachfolgenden Webseite erneut angezeigt werden, um sie überprüfen zu können. POST-Daten können aber auch ohne die aktive Beteiligung des Benutzers verwendet werden. Beispielsweise könnte die Zeit, die ein Nutzer auf einer Webseite verbracht hat, bevor er auf die nächste wechselt, übertragen werden. Basierend auf diesen Informationen könnte der Inhalt der aufgerufenen Webseite personalisiert werden.

Das Problem beim Speichern von Webseiten, das durch POST-Daten auftritt, ist, dass eine einzige URL als Input unzureichend ist. Um eine Kopie erstellen zu können, müsste die entsprechende URL inklusive der POST-Daten der vorhergehenden Webseite verfügbar sein. Beim Aufruf der URL müssten diese Daten dann an den HTTP-Request angehängt werden.

Dieses Problem ist nur teilweise lösbar. Gelegentlich kann basierend auf dem HTML-Code einer Webseite darauf rückgeschlossen werden, welche Variablennamen mit welchen Werten übermittelt werden, wie es beispielsweise bei Formularen der Fall ist. Codestück 17 zeigt ein solches Formular.

```
<form action="/user" method="post" id="user-login" accept-charset="UTF-8">
       <input type="text" id="edit-name" name="name" value="" size="60" maxLength="60"/>
       <input type="password" id="edit-pass" name="pass" size"60" maxlength="128"/>
       <input type="submit" id="edit-submit" name="op" value="Anmelden"/>
</form>
```

Codestück 17: Formular mit POST-Daten

Bei dem Formular aus Codestück 17 handelt es sich um einen simplen Login, welcher die Namen der POST-Variablen festlegt. Falls die Namen der zu übertragenden Variablen vom Server nicht verändert werden, so kann durch die Übermittlung der entsprechenden Variablen und ihren Werten die Darstellung der Webseite rekonstruiert werden.

11.1.3 Sessions

Bisher wurde in diesem Kapitel das Wort Benutzersitzung mehrmals erwähnt, dabei ging es darum zu unterscheiden, ob sich der visualisierte Inhalt einer Webseite durch die Interaktion mit dem Benutzer verändert oder ob er unabhängig davon ist.

Eine Session ist nun eine Benutzersitzung, die während der gesamten Dauer des Besuches der Webseite aktiv ist. Dies bedeutet, dass der Inhalt der angezeigten Webseite abhängig von einer Identifikationsnummer ist, die bei jedem Webseitenaufruf übermittelt wird. Die hierfür verwendeten Methoden sind Cookies sowie POST-, und GET-Daten.

Um Sessions in Webstamp zu implementieren, müssten alle Methoden, die zur Übertragung dieser ID verwendet werden, unterstützt werden. Die Situation, die bei Cookies und POST-Daten auftritt, wurde in den beiden vorhergegangenen Unterkapiteln bereits erörtert. Bei der Übermittlung mittels GET-Daten gibt es keine Probleme, da die ID an die URL mit Parametern angehängt wird.

Eine weitere Eigenschaft von Sessions ist ihre kurze Laufzeit. Dies bedeutet, dass eine Session-ID nur für einen begrenzten Zeitraum gültig ist, üblicherweise eine halbe Stunde, und danach nicht mehr verwendet werden kann. Trotz Übertragung einer Session-ID kann es nun vorkommen, dass eine Webseite zu zwei verschiedenen Zeitpunkten zwei verschiedene Ergebnisse liefert, dies liegt daran, dass die ID abgelaufen ist.

Das Speichern von Webseiten, die auf Sessions basieren, sieht man von den Problemen mit POST-Daten ab, ist teilweise realisierbar, allerdings muss der Sicherungsaufruf in dem Zeitfenster erfolgen, in dem die Session gültig ist. Da dieses Zeitfenster normalerweise bei jedem Aufruf zurückgesetzt wird, sollte dies keine Schwierigkeit darstellen.

11.1.4 JavaScript (AJAX)

Sehr viele Webseiten verwenden Skripte, um ihre Webseite für den Benutzer ansprechender und intuitiver zu gestalten. Mittels AJAX (Asynchronem-JavaScript) können beispielsweise interaktive Menüs gestaltet werden. Hierbei werden vorhandene Technologien so miteinander kombiniert, dass ein größerer Nutzen als nur die Summe ihrer Teile entsteht. Die folgenden Technologien sind von AJAX betroffen:

- Standardmäßige Präsentation von Inhalten mittels XHTML und CSS
- Dynamische Darstellung und Interaktion durch die Benutzung des Document Object Models
- Austausch von Daten und deren Manipulation mittels XML und XSLT
- Asynchrones Sammeln von Daten mittels XMLHttpRequests
- JavaScript, das dafür eingesetzt wird, die aufgelisteten Technologien miteinander zu verbinden [8]

Durch die Verwendung von AJAX im Web ergeben sich im Rahmen von Webstamp einige Probleme, da das Aussehen einer Webseite in einem solchen Fall nicht mehr an die URL der Webseite alleine gebunden ist, sondern auch an die Aktionen des Benutzers. Ohne die Benutzungshistorie in irgendeiner Form zu dokumentieren, kann Webstamp mit dem kontinuierlichen Nachladen Inhalten von nicht umgehen. Dies bedeutet, das Ergebnis einer AJAX-Benutzersitzung kann zurzeit nicht vollständig gespeichert werden. Abbildung 36 zeigt den Unterschied zwischen dem herkömmlichen Ansatz und dem AJAX-Ansatz.

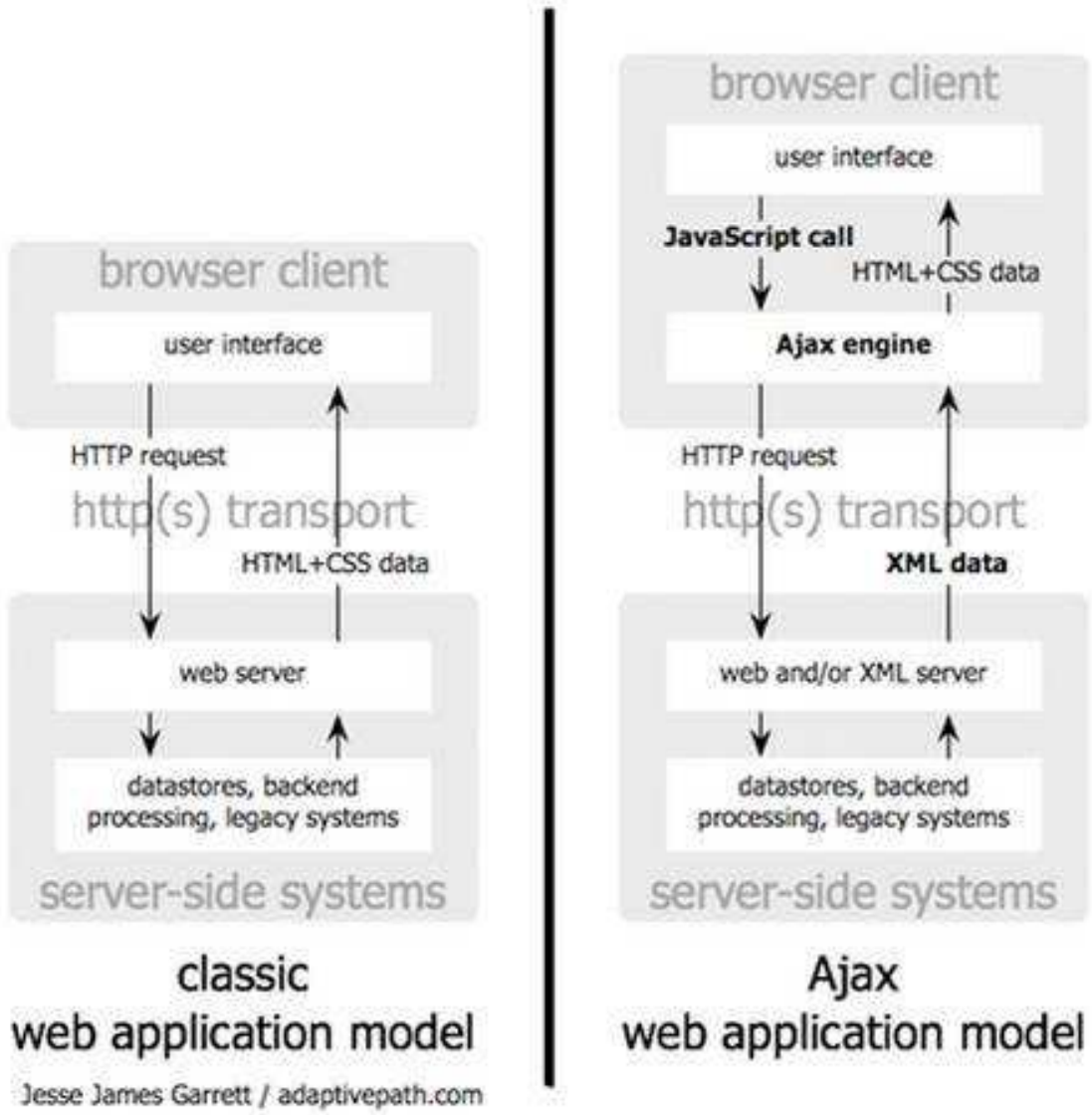

Abbildung 36: Vergleich des klassischen und des AJAX–Ansatzes

Das Sichern von Elementen, die zur Laufzeit nachgeladen werden, ist möglich, falls die Ressourcen mit ihrer kompletten URL im Code angegeben werden. Die Darstellung zum Zeitpunkt der Sicherung festzuhalten, wie es beispielsweise beim Öffnen eines AJAX Menüs der Fall ist, ist hingegen mit sehr großem Aufwand verbunden. Um dies realisieren zu können, müsste in der gesicherten Version jedes Skript automatisch so angepasst werden, dass es dem Zustand entspricht, den es zum gewünschten Betrachtungszeitraum einnahm.

Um diese Funktionalität in Webstamp implementieren zu können, müsste der komplette Zustand der Skripte zum Betrachtungszeitraum gesichert werden, um ihn später erneut einspielen zu können.

12 Literaturverzeichnis

[1] N.N., „Proxy Listen,“ [Online]. Available:

http://www.proxy-listen.de/Tutorials/Unterschiede-zwischen-Elite-Anonymen-und-Transparenten-Proxy-Servern.html. [Zugriff am 29 März 2012].

[2] S. Glassmann, „A Caching Relay for the World Wide Web,“ Elsevier Sience, Genf, 1994.

[3] S. I. P. Cao, „Cost-Aware WWW Proxy Caching Algorithms,“ USENIX Symposium, 1997.

[4] O. S. M. Rabinovich, „Web Caching and Replication,“ Sigmod Record, 2002.

[5] N.N., „RFC 3143,“ [Online]. Available: http://www.ietf.org/rfc/rfc3143.txt. [Zugriff am 8 Februar 2012].

[6] N.N., „Squid Cache,“ [Online]. Available: http://www.squid-cache.org. [Zugriff am 22 Februar 2011].

[7] N.N., „http://www.onb.ac.at/about/webarchivierung.htm,“ [Online].

Available: http://www.onb.ac.at/about/webarchivierung.htm. [Zugriff am 15 Mai 2012].

[8] N.N., „MMCCauley,“ [Online].

Available: http://mmccauley.net/com601/material/ajax_garrett.pdf. [Zugriff am 19 April 2012].

[9] M. Sonntag, „Der Nachweis von Inhalten im Internet,“ in s Transformation juristischer Sprachen, Wien, OCG 2012, 2012, pp. 131-139.

[10] I. B. H. Appel, Ein sicherer, robuster Zeitstempeldienst auf der Basis verteilter RSA-Signaturen., P. Horster: DuD Fachbeiträge, 2000.

[11] S. B. R. M. a. T. W. L. Page, „The PageRank Citation Ranking: Bringing Order to the Web," Stanford Digital Libary Technologies Project, 1998.

[12] P. Lawrence, „Method for node ranking in a linked database". US Patent US6285999, 4 September 2001.

[13] Haughton Mifflin Company, The American Heritage Dictionary of the English Language, Haughton Mifflin Company, 2009.

[14] „Java Forum," 2011. [Online]. Available: http://www.java-forum.org/.

[15] „Stackoverflow," 2011. [Online]. Available: http://stackoverflow.com.

[16] „Web Archive," 2011. [Online]. Available: http://web.archive.org.

[17] N.N., „RFC 3261," Juni 2002. [Online].
Available: http://tools.ietf.org/html/rfc3261. [Zugriff am 2011].

[18] S. M. K. Q. Wasim Ahmad Bhat, „Efficient Handling of Large Storage: A Comparative Study of Some Disk File Systems," INDIACom-2011, 2011.

Markus Haudum

geboren am 6.4.1985 in Linz

Ferihumerstraße 35, 4040 Linz

markus.haudum@gmail.com

0676/9059251

Aktuelle Tätigkeit

Informatik Student an der JKU Linz.

Masterstudium mit Kernfach „Netzwerke und Sicherheit“ und Nebenfach „Software Engineering“,

Ausbildung

Seit 2009	Masterstudium mit Kernfach „Netzwerke und Sicherheit“ und Nebenfach „Software Engineering“, JKU Linz.
2005 – 2009	Bachelorstudium Informatik, JKU Linz
1999 – 2004	BORG Linz, Schwerpunkt Informatik
1995 – 1999	Hauptschule, Auhof
1991 – 1995	Volksschule, Weberschule

2004 – 2005 **Präsenzdienst**

Zivildienst im BBRZ

Berufserfahrung

Ferialjobs

2002	Schachermayer, Lager
2006	Rosenbauer International AG; Informatik
2007	Rosenbauer International AG, Informatik
2008	Rosenbauer International AG, Informatik

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Masterarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Die vorliegende Masterarbeit ist mit dem elektronisch übermittelten Textdokument identisch.