

Technische Universität München
Fakultät für Informatik
Lehrstuhl für Rechnerarchitektur

Systementwicklungsprojekt

Entwurf und Implementierung einer Suchmaschine für
Konferenzankündigungen im Internet

Bearbeiter: Christoph Mühlich
Aufgabensteller: Prof. Arndt Bode
Betreuer: Dr. Max Walter

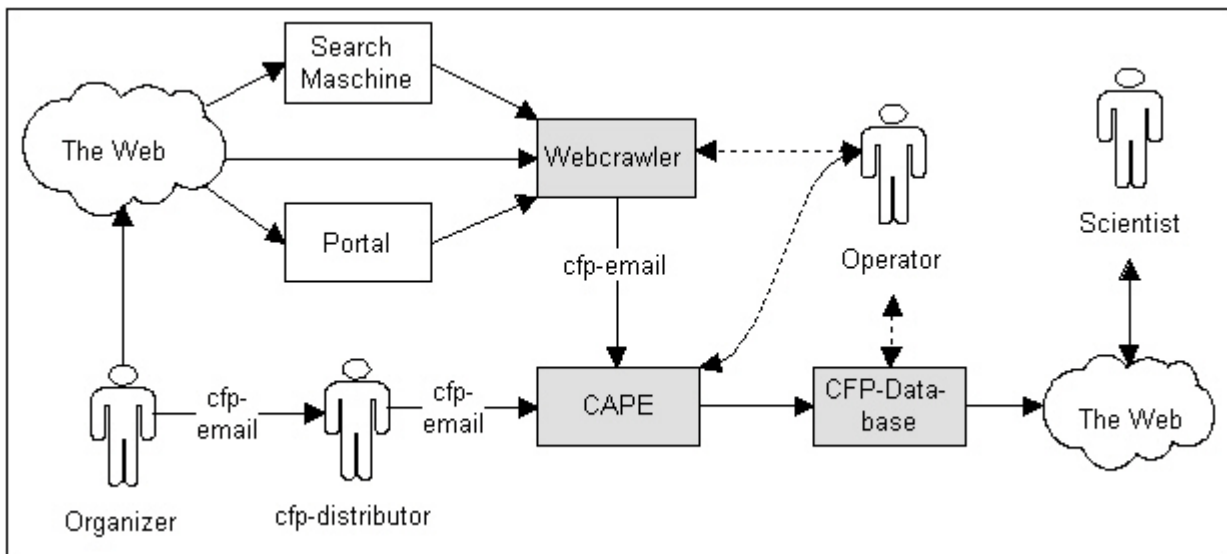
Garching, 17. März 2004

1. Aufgabenstellung

Dieser Abschnitt behandelt die Aufgabenstellung dieses Systementwicklungsprojektes und betrachtet die sich daraus ergebenden Ansätze zur Lösung. Die tatsächliche Lösung wird dann im nächsten Abschnitt ausführlicher erläutert.

1.1. Problembeschreibung

Im Rahmen dieses Systementwicklungsprojektes soll ein Programmpaket zur maschinellen Verarbeitung von Konferenzankündigungen, sog. „Calls for papers“ (im folgenden „CFP“ genannt), entwickelt werden. Dieses Programmpaket setzt sich aus drei verschiedenen Programmen zusammen. Die Programme zum automatischen Erkennen und Verarbeiten von Mails, die CFPs enthalten, sowie die Verwaltung und Katalogisierung der CFPs in einer Datenbank sind von anderen Studenten im Rahmen des SEPs entwickelt worden. Die folgende Grafik zeigt den prinzipiellen Aufbau des Programmpaketes. Die grau markierten Teile stellen dabei die Teilprojekte dieses SEPs dar.



In diesem Bild bezeichnet „Webcrawler“ das Programm zur vorliegenden Dokumentation, „CAPE“ ist das Teilprojekt „Computer Aided Processing of Email“ zum automatischen Erkennen von CFP-Mails und die „CFP-Databse“ mit dem Webzugriff stellt das Teilprojekt zur Datenhaltung und Abfrage dar.

Das dritte Teilprogramm, um welches es in dieser Dokumentation gehen soll, ist der so genannte Webcrawler, der die Aufgabe hat, das Internet aktiv nach CFPs zu durchsuchen. Dabei soll es sich (ausgehend von einer oder mehreren Startseiten) anhand der Hyperlinks durch das Internet vorarbeiten und Seiten, die einen CFP enthalten, per Mail an die zuvor erstellten Programme senden.

Die Kriterien, nach denen dabei vorgegangen wird, sollen flexibel einstellbar sein, damit das Programm an die Bedürfnisse angepasst werden kann. So soll es z.B. möglich sein, nur nach CFPs zu einem bestimmten wissenschaftlichen Bereich zu suchen. Das Programm als solches soll unbeaufsichtigt und ohne Interaktion mit dem Benutzer laufen. Es ist vorgesehen, dass das Programm nachts selbsttätig die Suche durchführt, am Morgen anhält und am Abend die Suche fortsetzt.

1.2. Lösungsansätze

Als erster Ansatz zur Lösung der Aufgabe liegt die Entwicklung eines sog. Crawlers nahe. Dieser muss in der Lage sein zu erkennen, welche Seiten er schon besucht hat und welche noch zu besuchen sind. Beim Durchlaufen der gefundenen Hyperlinks besteht zum einen die Möglichkeit, dies als Tiefensuche durchzuführen (d.h. der erste gefundene Hyperlink auf einer Seite wird sofort analysiert), zum anderen kann eine Breitensuche durchgeführt werden, bei der alle Seiten auf einer Ebene analysiert werden, bevor zur nächst tieferen Ebene gewechselt wird.

Bei der Analyse der einzelnen Seiten muss einerseits nach Kriterien für einen CFP gesucht werden, andererseits müssen Hyperlinks erkannt und zur nachfolgenden Analyse gespeichert werden.

Für die Bestimmung einer Seite als CFP gibt es zwei Möglichkeiten. Als erste Möglichkeit kann man definieren, dass dann, wenn ein Kriterium erfüllt ist, die Seite als CFP bestimmt wird. Dabei sind die einzelnen Kriterien unabhängig voneinander. Als zweite Möglichkeit kann jedem Kriterium ein Wert zugewiesen werden, der dann, wenn das Kriterium erfüllt ist, dem Wert der Seite hinzuaddiert wird. Übersteigt der Wert der Seite eine einstellbare Schwelle, so wird die Seite als CFP betrachtet. Bei dieser Möglichkeit sind die Kriterien nicht unabhängig voneinander, da es durchaus vorkommen kann (und im Normalfall sogar gewollt ist), dass ein Kriterium alleine nicht ausreicht, um einen CFP zu erkennen, mehrere zusammen allerdings schon.

Auch die Kriterien an sich sind unterschiedlich. So muss es möglich sein, nicht nur nach einem bestimmten Text innerhalb der Seite zu suchen, sondern auch die Struktur der Seite zu beachten. Deshalb sollte es auch möglich sein, den Typ der Seite (also z.B. eine HTML-Datei oder eine Textdatei) in die Kriterien einzubeziehen. Ebenso ist es vorgesehen, nach einem Datum innerhalb der Seite zu suchen und die Angaben im Header einer HTML-Datei speziell zu gewichten.

Im Rahmen der Realisierung wurde auf die angesprochenen Lösungsansätze eingegangen und eine dementsprechende Implementierung vorgenommen. Die dabei getroffenen Entscheidungen werden an der entsprechenden Stelle erläutert, sofern es erforderlich erscheint.

2. Realisierung

Für die Realisierung wurden die Lösungsansätze aus 1.2 betrachtet und daraus ein Konzept erstellt, um die Anforderung möglichst optimal zu erfüllen.

Eine der Anforderungen war es, dass die Suche weitestgehend konfigurierbar und anpassbar sein soll. Daraus ergab sich die Notwendigkeit, die Suchkriterien nicht in das Programm aufzunehmen, sondern eine Möglichkeit bereitzustellen, diese Kriterien auf flexible Weise angeben zu können. Dies wurde durch die Auslagerung der Definition der Kriterien in sog. Kriteriendateien erreicht. Diese Dateien werden vom CARACAS-Webcrawler beim Start der Suche eingelesen und verarbeitet. Eine Beschreibung der Syntax dieser Dateien findet sich im Anhang 5.3.

Des Weiteren erfolgt die Allgemeine Konfiguration des Programms über eine Konfigurationsdatei, die vom CARACAS-Webcrawler ebenfalls beim Start der Suche eingelesen wird. Diese Datei enthält zum einen allgemeine Angaben wie z.B. die Uhrzeit, zu der das Programm die Suche durchführen soll, zum anderen aber auch die URLs, von denen aus die Suche begonnen werden soll. Eine genaue Beschreibung dieser Datei befindet sich im Anhang unter dem Punkt 5.2.

2.1. Entwicklungs- und Laufzeitumgebung

Der CARACAS-Webcrawler wurde mit Visual Basic Professional 6.0 unter Windows XP erstellt. Zur Laufzeit benötigt das Programm die Visual Basic Laufzeit-Bibliothek sowie einige weitere Dateien. Eine genaue Auflistung dieser Dateien befindet sich im Anhang. Zum Mailversand muss das Programm „Blat.exe“ im Programmverzeichnis verfügbar sein. Alle benötigten Dateien und Bibliotheken werden bei der Installation des Programms automatisch kopiert und registriert.

Nach der Installation müssen die Konfigurationsdateien im Programmverzeichnis an die konkreten Bedürfnisse angepasst werden. Eine Beschreibung dieser Dateien sowie eine Dokumentation der Programmoberfläche finden sich im Anhang.

2.2. Klassen- und Objektbeschreibung

CARACAS-Webcrawler ist objektorientiert aufgebaut. Dadurch ist eine klare Trennung der einzelnen Aufgaben innerhalb des Programms möglich. Auch eine spätere Änderung oder Erweiterung des Programms wird dadurch unterstützt. In den folgenden Abschnitten werden die verwendeten Klassen und Objekte beschrieben. Ebenso wird auf das Zusammenspiel der Objekte eingegangen.

2.2.1. Link-Manager

Das zentrale Element des CARACAS-Webcrawler ist der sog. Link-Manager. Dieser verwaltet alle URLs, mit denen das Programm zu tun hat. Beim Start der Suche werden die in der Konfigurationsdatei angegebenen URLs dem Link-Manager hinzugefügt, ebenso die während der Suche gefundenen URLs. Der Link-Manager gibt sie dann auf Anfrage eines Link-Parsers aus und vermerkt die URL als „*Requested*“.

Der Link-Manager ist auch dafür zuständig, doppelte Links auszusondern. Sobald eine neue URL dem Link-Manager hinzugefügt wird, prüft dieser, ob der entsprechende Link schon in der internen Liste vorhanden ist. Ist dies nicht der Fall, wird die URL aufgenommen und erhält den Status „*Unvisited*“. Ansonsten wird die URL verworfen.

Beim Hinzufügen einer neuen URL wird vom Link-Manager auch geprüft, ob die Tiefe der URL kleiner als die in der Konfigurationsdatei angegebene maximale Suchtiefe ist. Zu diesem Zweck wird bei jeder URL vermerkt, auf welcher Tiefe (ausgehend von den beim Start angegebenen

URLs) sie sich befindet. Die Tiefe einer neuen URL bestimmt sich dann aus der Tiefe der URL, durch welche die neue URL gefunden wurde zuzüglich 1. Übersteigt die Tiefe einer neuen URL die maximale Suchtiefe, so wird die neue URL vom Link-Manager abgelehnt, unabhängig davon, ob die URL schon besucht wurde oder nicht.

Durch die Implementierung des Link-Managers wird festgelegt, ob die Suche als Tiefensuche oder als Breitensuche erfolgen soll (Siehe Lösungsansätze). In der vorliegenden Implementierung wurde eine Breitensuche verwendet. Zum einen ist in diesem Fall die Verwaltung der URLs sehr einfach, sie kann mit einfachen Listen erfolgen. Eine neue URL wird dann einfach an das Ende der Liste angefügt. Die Abarbeitung der Liste erfolgt in der Reihenfolge, in der die URLs hinzugefügt wurden. Zum anderen ist kann es auf diese Weise nicht so leicht zu einem „Totlaufen“ der Suche kommen, indem es vermieden wird, durch den ersten Link einer Seite in einen Bereich des Internets zu gelangen, der keine CFPs enthält. Natürlich wird dieser Bereich immer irgendwann erreicht werden, doch der Crawler bearbeitet diesen Bereich nicht auf einmal komplett (und würde dadurch für lange Zeit keine Ergebnisse liefern), sondern wechselt immer zwischen verschiedenen Bereichen. Als Beispiel sei hier eine Seite genannt, die z.B. Links auf eine Sammlung von CFPs enthält. Bei der Tiefensuche würde der Crawler den ersten Link bearbeiten, somit den ersten CFP erkennen und dann die Links in diesem CFP verfolgen. Die anderen CFPs in der Sammlung würden vorerst nicht bearbeitet werden, obwohl die Seite der Sammlung schon gelesen wurde. Je nach Anzahl und weiterführender Tiefe der Links auf der ersten CFP-Seite kann eine lange Zeit vergehen, bis die restlichen CFPs der Sammlung verarbeitet werden.

2.2.2. Link-Parser

Die eigentliche Suche nach CFPs findet im so genannten Link-Parser statt. Der Link-Parser fragt beim Link-Manager an, ob es noch nicht besuchte Links gibt. Wenn dies der Fall ist, so holt sich der Link-Parser die URL und verarbeitet diese. Die URL erhält dabei den Status „*Requested*“.

Der Link-Parser versucht als erstes, die durch die URL angegebene Seite aus dem Internet zu laden. Dabei wird eine Timeout-Zeit eingestellt, die in der Konfigurationsdatei angegeben wurde. Konnte die Datei nicht geladen werden (sei es, weil sie nicht mehr vorhanden ist oder der Server die Seite nicht in der geforderten Zeit ausliefern konnte), wird die URL als „*Erroneous*“ gekennzeichnet. Konnte die Datei geladen werden, so erfolgt die Suche innerhalb der Datei. Als erstes wird dabei nach weiteren URLs in der Datei gesucht. Es werden sowohl absolute als auch relative Verweise gefunden und verarbeitet. Bei der vorliegenden Implementierung werden nur HTML-Verweise innerhalb von <HREF>-Tags gesucht. Verweise, die innerhalb von Script-Anweisungen oder nur im Klartext vorhanden sind, werden noch nicht verarbeitet. Alles gefundenen URLs werden an den Link-Manager weitergereicht, der die weitere Analyse und Verarbeitung der URLs verwaltet.

Anschließend beginnt die Suche nach den Kriterien innerhalb der Datei. Dazu wird für die Seite der aktuelle Punktestand mit 0 initialisiert. Um sowohl eine Case-Sensitive als auch eine nicht Case-Sensitive Suche durchführen zu können, wird der Inhalt der Seite einmal in der Originalfassung und einmal komplett in Kleinbuchstaben gespeichert.

Nun werden alle Kriterien in der Reihenfolge der Definition innerhalb der Kriteriendatei abgearbeitet. Ist ein Kriterium erfüllt, d.h. der angegebene Suchtext wurde in der Seite gefunden, so wird der Punktestand der Seite um das Gewicht des Kriteriums erhöht bzw. verringert. Da Kriterien auch ein negatives Gewicht haben können, werden immer alle Kriterien geprüft. Erst dann wird der Endpunktestand der Seite mit dem festgelegten Schwellenwert verglichen. Ist der Punktestand größer oder gleich dem Schwellenwert, so wird die bearbeitete Seite als CFP erkannt und entsprechend verarbeitet. Dazu wird ein CFP-Objekt angelegt und mit den entsprechenden Daten gefüllt. Die weitere Verarbeitung, wie z.B. Formatierungen oder den Mail-Versand, übernimmt dann das CFP-Objekt.

2.2.3. Link-Objekt

Das Link-Objekt dient nur zur Speicherung aller notwendigen Angaben zur Verwaltung der URLs. Es enthält die eigentliche URL, den Basispfad, den Status und die Tiefe des Links. Der Status kann einen der folgenden Zustände annehmen:

Zustand	Beschreibung
cmUnvisited	Der Link wurde noch nicht besucht
cmRequested	Der Link wurde einem Link-Parser übergeben
cmVisited	Der Link wurde verarbeitet
cmErroneous	Beim Lesen des Links trat ein Fehler auf

Der Basispfad eines Link-Objekts ist die vollständige, absolute URL ohne den Namen der Seite selbst.

Das Link-Objekt beinhaltet keine Funktionalität (außer der automatischen Ermittlung des Basispfades) und stellt deshalb keine Methoden bereit.

2.2.4. Kriterien

Die einzelnen Kriterien, die für die Suche in der Kriteriendatei definiert wurden, werden in Kriterienobjekten verwaltet. Diese Objekte stellen, ähnlich dem Link-Objekt, eine Hülle zur Datenspeicherung dar.

In einem Kriterienobjekt werden die Daten eines Kriteriums verwaltet. Dazu gehört als erstes der Typ des Kriteriums. Dieser Typ gibt an, welche Art von Suche innerhalb der HTML-Seite durchgeführt werden soll. Folgende Typen stehen dabei zur Auswahl:

Typ	Beschreibung
cmFileType	Es wird der Dateityp der Seite geprüft
cmSimpleText	Das angegebene Stichwort wird gesucht
cmNear	Es wird geprüft, ob die beiden angegebenen Stichwörter nahe beieinander stehen
cmContainsDate	Es wird geprüft, ob die Seite ein Datum enthält
cmHeader	Das angegebene Stichwort wird im HTML-Header gesucht
cmAND	Alle angegebenen Stichwörter müssen in der Seite vorkommen
cmOR	Mindestens eines der angegebenen Stichwörter muss in der Seite vorkommen

Je nach ausgewähltem Typ der Suche können mehrere Suchbegriffe angegeben werden. Mit der Einstellung „*CaseSensitive*“ wird festgelegt, ob der Suchbegriff genauso in der Seite vorkommen muss, wie er in der Kriteriendatei angegeben ist, oder ob bei der Suche die Groß- und Kleinschreibung nicht beachtet werden soll.

Für den Suchtyp „*cmNear*“ wird noch ein Parameter benötigt, der angibt, wie weit zwei Worte voneinander entfernt sein dürfen, um noch als „nahe beieinander stehend“ zu gelten. Der Abstand zwischen zwei Worten ist dabei die Anzahl Zeichen, die zwischen dem Ende des ersten Wortes und dem Anfang des zweiten Wortes stehen. Als Standard ist ein Abstand von 50 Zeichen eingestellt.

Weiterhin wird im Kriterienobjekt noch das Gewicht des Kriteriums gespeichert. Dieses Gewicht wird zur Punktzahl der Seite addiert, wenn das Kriterium erfüllt ist. Zu beachten ist, dass Gewichte auch negativ sein können. Dadurch ist es möglich, bestimmte Stichwörter auszuschließen, bzw. festzulegen, dass bestimmte Stichwörter eher gegen einen CFP sprechen. Gewichte können einen Wert haben zwischen -32.768 bis 32.767 annehmen.

2.2.5. CFP-Objekte

Wenn nach der Prüfung einer Seite durch den Link-Parser der Punktestand der Seite größer oder gleich dem Schwellenwert ist, wird ein CFP-Objekt für diese Seite angelegt. Dieses Objekt enthält die URL der Seite, den Inhalt und die erreichte Punktezahl. Die weitere Verarbeitung der Seite übernimmt dann das CFP-Objekt.

Die Aufgabe des CFP-Objektes ist es, den Text der HTML-Seite so zu formatieren, dass er in einer Mail versendet und vom CAPE-Erkenner bearbeitet werden kann. Dazu werden einige grundlegende Strukturierungen von HTML umgewandelt in passende Elemente eines ASCII-Textes und alle weiteren HTML-Elemente werden gelöscht.

Tabellenzeilen (<tr>) werden in einen Zeilenumbruch geändert, Tabellenspalten (<td>) werden durch Tabulatoren ersetzt. Dadurch wird erreicht, dass der Text der Seite besser von CAPE-Erkenner erkannt werden kann.

Im nächsten Schritt werden alle noch verbliebenen HTML-Elemente aus dem Text gelöscht, so dass nur der eigentliche Inhalt der Seite übrig bleibt. Es folgen noch einige Aufräumarbeiten wie z.B. das Löschen von doppelten Zeilenumbrüchen.

Nach der Formatierung des Ausgabertextes wird das CFP-Objekt wieder an den Link-Parser zurückgegeben, der dann den Versand per Mail einleitet.

2.2.6. Das Mailer-Objekt

Um nach der Erkennung eines CFPs diesen per Mail zu versenden, wird das Mailer-Objekt verwendet. Dieses greift auf das externe Programm Blat.exe zurück. Blat.exe ist ein Programm, das es ermöglicht, Mails auf der Kommandozeile per SMTP zu versenden. Durch die Verwendung dieses Programms war es möglich, auf die Entwicklung einer eigenen SMTP-Engine zu verzichten.

Blat.exe erwartet als Parameter den Empfänger, die Angaben zum SMTP-Server und SMTP-Postfach, sowie die Datei, die den Text der Mail enthält. Dazu erzeugt das Mailer-Objekt zuerst eine Textdatei im Programmverzeichnis mit dem Namen „mail_body.txt“. Diese Datei enthält den kompletten Text des gefundenen CFP sowie die URL der Ursprungsseite (sofern das in der Konfiguration so eingestellt wurde). Die Datei wird dann an Blat.exe übergeben, welche dann die Versendung übernimmt.

Um einen Konflikt durch mehrfaches anmelden an der Mailbox durch Blat.exe zu vermeiden, wird Blat.exe in einem DOS-Fenster gestartet und das Mailer-Objekt wartet auf die Beendigung des DOS-Fensters. Durch dieses Verhalten und durch die Tatsache, dass immer nur ein Mailer-Objekt vorhanden ist, wird sichergestellt, dass niemals mehrere Mails gleichzeitig versendet werden.

Als SMTP-Server kann jeder beliebige Server verwendet werden, auf dem eine gültige Mailbox vorhanden ist und der aus dem aktuellen Netz erreichbar ist.

3. Test

Das Programm wurde während und nach der Implementierung verschiedenen Tests unterzogen. Diese Tests umfassten das allgemeine Verhalten des Programms als auch spezielle Tests der Suche. Insbesondere wurden die Suchergebnisse getestet, um feststellen zu können, wie die Kriterien aufgebaut sein müssen, um optimale Suchergebnisse zu erhalten.

3.1. Allgemeine Tests

Die allgemeinen Tests umfassten zum einen die bei der Programmerstellung üblichen Tests auf Korrektheit der verwendeten Algorithmen. So wurde z.B. getestet, ob die Suchfunktionen einen tatsächlich enthaltenen Suchbegriff auch wirklich finden bzw. einen nicht enthaltenen Begriff nicht doch fälschlicherweise als Treffer werten. Dies wurde mit Hilfe mehrere lokaler HTML-Seiten und unterschiedlichen Konfigurationen getestet. Durch manuelle Abfrage des Punktstands einer Seite während der Suche konnte so für jeden Suchbegriff der Konfiguration festgestellt werden, ob das Programm den Begriff gefunden hat oder nicht.

Weiterhin wurde getestet, ob das Programm die geforderten Ruhezeiten einhält. Dazu wurde ein Zeitfenster angegeben, in dem das Programm die Suche durchführen soll. Nach dem Programmstart wurde dann die Systemuhr umgestellt, um verschiedene Zeiten zu simulieren. Das Programm musste nun entsprechend der eingestellten Uhrzeit seine Arbeit unterbrechen bzw. an der unterbrochenen Stelle fortsetzen.

Während der Tests wurde weiterhin die durchschnittliche Prozessorauslastung des Programms ermittelt. Dazu wurde die im Taskmanager angegebene CPU-Zeit durch die gesamte Programmlaufzeit geteilt. Als Mittelwert mehrerer Messungen ergab sich eine Prozessorauslastung von 21,3 %.

3.2. Test auf „False positive“

Der Test auf „False positive“ soll feststellen, ob das Programm eine Seite als CFP erkennt, obwohl es kein CFP ist. Dieser Test ist natürlich sehr stark von den gewählten Suchbegriffen abhängig. Im allgemeinen Fall (der hier auch getestet wurde) sollen alle Calls for Papers, egal aus welchem Fachgebiet, gefunden werden. Dazu ist es nur notwendig, die allgemeinen Kriterien für den Inhalt eines CFP zu kennen. Wenn hingegen nur nach speziellen Fachgebieten gesucht werden soll, so müssen den allgemeinen Kriterien noch fachspezifische hinzugefügt werden. Diese Kriterien dann so zu wählen, dass sie einerseits allgemein genug sind, um keine CFPs aus dem Fachgebiet zu übersehen, andererseits aber speziell genug, um keine CFPs aus anderen Fachbereich zu finden, gestaltet sich schwierig. In diesem Test wurde daher nur der allgemeine Fall getestet.

Der konkrete Test hatte als Startseiten die Homepage der Fakultät für Informatik an der TU München (<http://www.informatik.tu-muenchen.de/index.html>), die Homepage des Heise-Verlages (<http://www.heise.de/>) und die private Seite des Autors (<http://www.christoph-muehlich.de/>) sowie eine Seite, die eine Liste von CFPs enthält (<http://dutetvg.et.tudelft.nl/~alex/CFP/>). Die Suchtiefe wurde nicht begrenzt. Die verwendeten Kriterien finden sich im Anhang unter „Kriterien_FP.txt“.

Nach über 7000 gelesenen Seiten wurde der Test abgebrochen und die Ergebnisse analysiert. Die als CFP erkannten Seiten dazu von Hand kontrolliert. Dabei ergab sich folgende Tabelle:

Seiten angefordert	7615
Seiten fehlerhaft/nicht verfügbar	102
Erkannte CFPs	259
Davon falsch erkannt	5

Die falsch erkannten Seiten waren dabei immer Linklisten zu richtigen CFPs. Diese Linklisten enthalten fast alle Merkmale eines CFPs, wie z.B. Datumsangaben und den Text „CFP“. Es ergibt sich aus der Gegenüberstellung der gefundenen CFPs und der davon falsch erkannten Seiten ein prozentueller Fehler von 1,93%.

3.3. *Test auf „False negative“*

Dieser Test soll zeigen, dass das Programm keine CFPs fälschlicherweise übersieht. Auch für diesen Test gilt das oben Gesagte bezüglich der speziellen Kriterien. Wenn ein Kriterium zwingend gefordert ist (im Sinne von: es hat ein hohes Gewicht), so kann eine Seite, die dieses Kriterium nicht enthält, aber trotzdem ein CFP ist, vom Programm nicht erkannt werden. Daher sollten die Kriterien relative geringe Gewichte haben, im Gegenzug sollten es aber für eine spezielle Suche viele Kriterien sein. In der Summe ergibt sich dann auch wieder der Schwellenwert, aber ein einzelnes Kriterium fällt nicht so stark ins Gewicht, wenn es nicht vorhanden ist. Auch für diesen Test wurde der allgemeine Fall verwendet.

Um den Test durchzuführen, wurde lokal eine Liste von CFPs mit den zugehörigen HTML-Seiten gespeichert. Diese Liste wurde geladen von der Seite <http://dutetvg.et.tudelft.nl/~alex/CFP/>, mit Stand vom 30.10.2003. Die Suchtiefe betrug 1, um ausgehend von der Indexseite, die Liste der CFPs nicht zu verlassen. Die verwendeten Kriterien finden sich im Anhang.

Die Analyse des Tests ergab folgende Tabelle:

Seiten angefordert	198
Davon korrekte CFPs	194
Seiten fehlerhaft/nicht verfügbar	0
Erkannte CFPs	198
Davon falsch erkannt	4

Auch bei diesem Test waren die fehlerhaften Seiten Linklisten zu weitere CFPs. Zu bemerken ist in diesem Fall, dass alle tatsächlich vorhandenen CFPs erkannt wurden.

4. Zusammenfassung / Ausblick

Insgesamt betrachtet kann gesagt werden, dass der CARACAS-Webcrawler die eingangs erwähnten Anforderungen erfüllt. Sowohl die automatische, unbeaufsichtigte Suche als auch der Zeitgesteuerte Ablauf wurden berücksichtigt. Ob das Programm allerdings die grundlegende Funktion, die Suche nach CFPs, erfüllt, hängt sehr stark von den definierten Kriterien ab. Sind es zu wenige oder zu allgemeine, wird der CAPE-Erkennen mit falsch erkannten CFPs überschwemmt. Sind die Kriterien hingegen zu speziell, so wird das Programm nur einen Bruchteil der gewünschten und vorhandenen CFPs finden. Beide Punkte können durch die Anpassung der Konfiguration gegeneinander abgewogen werden.

Im Laufe der Zeit wird sich eine Konfiguration finden, die genau das erfüllt, was man von ihr erwartet. Allerdings müssen die Kriterien immer wieder überprüft werden, da das Internet bekanntermaßen ein schnelllebiges Medium ist, in dem sich gebräuchliche Ausdrücke durchaus auch mal ändern können.

Der CARACAS-Webcrawler stellt ein geeignetes Hilfsmittel dar, um Inhalte im Internet zu finden. Er ist auch nicht auf Calls for Papers festgelegt. Um andere Inhalte zu finden, genügt es, die Konfiguration und die Kriterien zu ändern. So lässt sich nach allem suchen, was sich mit den bereitgestellten Suchmöglichkeiten eingrenzen lässt.

Im Laufe der genaueren Spezifikation und der Implementierung fielen weitere Punkte auf, die dieses Programm ergänzen bzw. erweitern könnten.

So wäre z.B. eine Datenbankanbindung denkbar, um alle schon besuchten Links einschließlich des Datums des letzten Besuchs abzuspeichern. Damit wäre es möglich, Seiten öfters zu besuchen, wenn eine bestimmte Zeit verstrichen ist. Ebenso wäre eine größere Unterstützung von Dateitypen denkbar, die es ermöglichen würde, z.B. auch dynamisch erzeugte Seiten zu verarbeiten oder PDF-Dokumente nach CPFs zu durchsuchen. Auch eine noch genauere Analyse der HTML-Struktur würde eine feinere Suche ermöglichen. So könnte man beispielsweise Stichwörter, die in Überschriften vorkommen, höher bewerten als andere. Oder auch die Anzahl an Hyperlinks, die zu einem Dokument führen, könnte als Kriterium dienen. Denkbar wäre auch eine Abfrage, ob bestimmte externe Inhalte wie z.B. Bilder in der Seite enthalten sind.

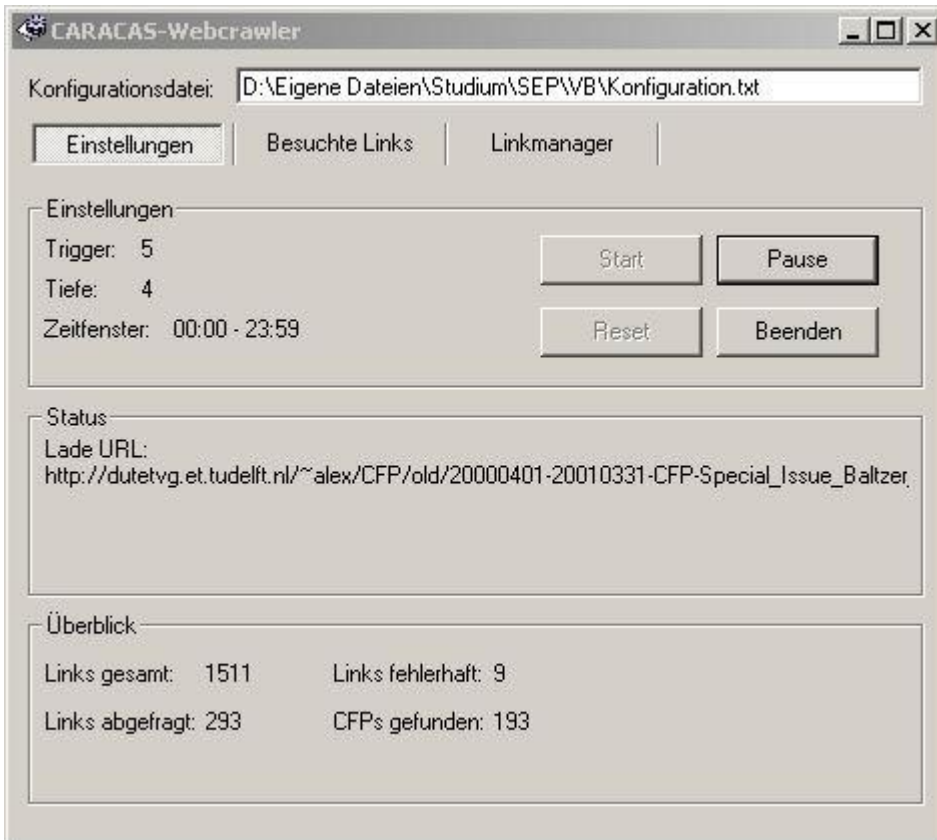
Ein weiterer denkbarer Punkt ist die Analyse des Kontexts, innerhalb dessen ein Link gefunden wurde. So könnte z.B. ein Link, der im Linktext schon den Begriff CFP enthält, bevorzugt überprüft werden oder ein Link der z.B. „Impressum“ enthält, könnte komplett verworfen werden.

Des Weiteren wäre durch es durch den objektorientierten Aufbau möglich, mehrere Link-Parser-Objekte zu verwenden. Dadurch könnten Leerlaufzeiten vermieden werden, die aufgrund des Wartens auf eine Server-Antwort entstehen können. Allerdings müssten dazu noch die Funktionen „Ad-dURL“ und „GetNextURL“ des Link-Managers entsprechend synchronisiert werden, damit die Verwaltung der gefundenen Links weiterhin konsistent bleibt.

5. Anhang

5.1. Handbuch zum Programm

Die meisten Einstellungen des Programms werden über die weiter unten erläuterten Dateien gesteuert. Daher enthält die Oberfläche des Programms relativ wenig Elemente. Das folgende Bild zeigt einen Screenshot des Programmfensters.



Das Eingabefeld „Konfigurationsdatei“ legt den Namen der Konfigurationsdatei fest, die das Programm verwenden soll. Die Schaltfläche „Start“ (hier deaktiviert, da das Programm gerade die Suche durchführt) startet die Suche gemäß der angegebenen Konfiguration. Die Schaltfläche „Pause“ veranlasst das Programm dazu, die Suche zu unterbrechen. Nach einem weiteren Klick auf diese Schaltfläche (die dann den Titel „Fortsetzen“ trägt) nimmt die Suche wieder auf. Mit einem Klick auf „Beenden“ lässt sich das Programm beenden. „Reset“ setzt alle Einstellungen des Programms zurück, einschließlich aller Daten im Linkmanager.

Über die Karteireiter am oberen Rand gelangt man zu verschiedenen Informationen des Programms. Der Reiter „Einstellungen“ zeigt das oben stehende Bild. Der Reiter „Besuchte Links“ zeigt eine Liste aller URL, die das Programm schon besucht hat, sowie den Punktstand, den eine Seite erhalten hat. Über den Reiter „Linkmanager“ erhält man eine Übersicht über die Links, die vom Programm erkannt worden sind einschließlich der Tiefe des Links.

Die Größe des Programmfensters lässt sich beliebig anpassen, unterhalb einer bestimmten Größe können jedoch nicht mehr alle Elemente angezeigt werden. Wenn das Programm minimiert wird, so verschwindet es aus der Taskleiste und erscheint nur noch als Symbol im Systemtray. Ein Doppelklick auf dieses Icon zeigt das Programmfenster wieder an.

5.2. Dokumentation der Konfigurationsdatei

Die Programmoptionen für den CARACAS-Webcrawler werden über eine Konfigurationsdatei angegeben. Dabei definiert jede Zeile eine Option. Kommentare sind Zeilen, die mit dem Prozentzeichen (%) beginnen. Kommentare sind nur als ganze Zeilen zulässig.

Es wird empfohlen, die Konfigurationsdatei im Programmverzeichnis abzulegen und ihr den Namen „Konfiguration.txt“ zu geben. Dieser Name ist im Programm eingestellt und wird nach dem Programmstart als erstes im entsprechenden Feld angezeigt. Es muss dann also kein neuer Name eingegeben werden.

Header der Datei:

Die erste Zeile der Datei muss wie folgt lauten:

```
CARACAS
```

Anhand dieses Eintrags überprüft das Programm, ob es sich um eine gültige Konfigurationsdatei handelt. Danach folgen die Einstellungen des Programms, wobei folgendes Format gilt:

```
<Feld> = <Wert>
```

Gültige Einträge für <Feld> und <Wert> listen die folgenden Tabellen auf:

Allgemeine Optionen

Feld	Beschreibung
Triggervalue	Legt den Schwellenwert fest, den eine Seite mindestens erreichen muss, um als CFP betrachtet zu werden. <Wert> muss zwischen 0 und 32.767 liegen. Der Standardwert ist 10.
MaxLinkDepth	Dieses Feld legt die maximale Tiefe fest, bis zu der Links verfolgt werden. <Wert> muss dabei zwischen 1 und 2.147.483.647 liegen. Wird für <Wert> -1 angegeben, so wird die Suche nicht beschränkt. Der Standardwert ist -1.
IncludeLink	Hiermit wird festgelegt, ob in der Mail, die nach dem Erkennen eines CFP versendet wird, die URL der Seite aufgenommen werden soll. Wenn <Wert> auf „N“ gesetzt wird, wird die URL nicht aufgenommen. Der Standardwert ist „Y“
TimeOut	Legt die Zeit in Sekunden fest, die bei der Anforderung einer Seite maximal verstreichen darf. Wenn innerhalb dieser Zeit keine Antwort des Servers erfolgt ist, wird das Laden der Seite abgebrochen und der Link als fehlerhaft vermerkt. <Wert> muss zwischen 1 und 32.767 liegen. Der Standardwert ist 5.
StartTime StopTime	Diese beiden Felder legen das Zeitfenster fest, innerhalb dessen die Suche stattfinden soll. Außerhalb dieser Zeit unterbricht das Programm die Suche. Beide Angabe müssen Uhrzeiten im Format mm:hh sein. Wenn die Felder nicht angegeben werden, so wird die Suche rund um die Uhr durchgeführt.
BlatLog	Wird diese Option auf „Y“ gesetzt, dann schreibt Blat.exe ein Log in die Datei Blat.log im Programmverzeichnis. Standardmäßig ist diese Option abgeschaltet.

Optionen zum Mailversand

Feld	Beschreibung
MailServer	Legt die Adresse des zu verwendenden SMTP-Servers fest
MailUser	Dieses Feld legt den Namen der Mailbox auf dem Server fest, die verwendet werden soll
MailLogin	Gibt den Benutzernamen zur oben angegebenen Mailbox an
MailPWD	Dieses Feld legt das zu MailLogin gehörende Passwort der Mailbox fest
MailSubject	Die Angabe in diesem Feld wird als Subject in jeder Mail verwendet. Dieses Feld ist optional. Der Standardwert lautet „CARACAS-Webcrawler“
Recipient	Mit diesem Feld wird die Empfängeradresse festgelegt

Alle oben aufgeführten Optionen zum Mailversand erwarten einen Text als Parameter und sind mit Ausnahme von „MailSubject“ immer erforderlich.

Nach der Angabe der Programmoptionen erfolgt die Angabe der Startseiten, die das Programm verwenden soll. Dazu muss als erstes folgende Zeile angegeben werden:

```
BeginLinks
```

Ab jetzt definiert jede folgende Zeile die komplette URL einer Startseite. Es können beliebig viele Seiten hinzugefügt werden. Den Abschluss der Auflistung der Startseiten bildet diese Zeile:

```
EndLinks
```

Im Anschluss an die Definition der Startseiten beginnt die Festlegung der zu verwendenden Kriteriendateien. Dazu muss als erstes diese Zeile angegeben werden:

```
BeginCriterias
```

Nun folgen in jeder Zeile die einzelnen Dateien, die verwendet werden sollen. Die Dateinamen müssen immer einschließlich ihres absoluten und vollständigen Pfades angegeben werden. Eine Ausnahme bilden Dateien, die im Programmverzeichnis liegen. Bei diesen genügt es, den Dateinamen anzugeben. Um die Liste der Kriteriendateien abzuschließen, dient diese Zeile:

```
EndCriterias
```

5.3. Dokumentation der Kriteriendatei

Eine Kriteriendatei ist eine normale Textdatei mit der Endung „.txt“. Jede Zeile wird separat eingelesen und verarbeitet. Kommentare sind Zeilen, die mit dem Prozentzeichen (%) beginnen. Kommentare sind nur als ganze Zeilen zulässig.

Header der Datei:

Die erste Zeile der Datei muss wie folgt lauten:

```
CARACAS
```

Anhand dieses Eintrags überprüft das Programm, ob es sich um eine gültige Kriteriendatei handelt. Es dürfen weder Leerzeilen noch Leerzeichen oder Tabulatoren vor dem Eintrag stehen.

Danach folgt die Definition der einzelnen Kriterien. Eine solche Definition ist wie folgt aufgebaut:

```
ENTRY
TYP <Suchart>
VAL <Wert>
CAS <Bool>
[PAR <Wert>]
<Text 1>
[ <Text 2>
...
<Text n>]
```

wobei folgende Optionen möglich sind:

- **TYP**
Mit diesem Feld wird festgelegt, welchen Typ die Suche haben soll. Für <Suchart> sind folgende Optionen möglich:

<Suchart>	Beschreibung
FileType	Es wird der Dateityp der Seite geprüft
Simple	Das angegebene Stichwort wird gesucht
Near	Es wird geprüft, ob die beiden Stichwörter nahe beieinander stehen
Date	Es wird geprüft, ob die Seite ein Datum enthält
Header	Das angegebene Stichwort wird im HTML-Header gesucht
AND	Alle angegebenen Stichwörter müssen in der Seite vorkommen
OR	Mindestens eines der angegebenen Stichwörter muss in der Seite vorkommen

- **VAL**
Diese Angabe legt das Gewicht des Kriteriums fest. Für <Wert> gilt folgendes:
-32.768 <= Wert <= 32.767

- **CAS**
Hiermit wird angegeben, ob die Suche unabhängig von der Groß- und Kleinschreibung sein soll.

<Bool>	Beschreibung
Y	Die Groß- und Kleinschreibung wird beachtet
N	Es findet keine Unterscheidung zwischen Groß- und Kleinschreibung statt

- **PAR**
Dieses optionale Feld legt beim Suchtyp „Near“ fest, wie groß der Abstand zwischen den angegebenen Worten maximal sein darf. Wird das Feld nicht angegeben, so wird der Standardwert 50 verwendet. Für <Wert> gilt folgendes:

$$0 < \text{Wert} \leq 32.767$$

- **Stichwortangaben**

Nach den oben beschriebenen Optionen folgt die Angabe der Texte, nach denen gesucht werden soll. Jede Zeile gibt dabei einen Text an, der sowohl aus nur einem Stichwort als auch aus mehreren Wörtern bestehen kann. Jede Zeile wird später exakt so gesucht, wie sie hier angegeben ist (mit Ausnahme der Groß- und Kleinschreibung, sofern die Option CAS den Wert N hat).

Abhängig davon, welche Suchart ausgewählt wurde, sind bestimmte Einschränkungen für die Stichwortangaben zu beachten:

Suchart	Einschränkung für Stichwörter
FileType	Es ist nur eine gültige Dateierweiterung zulässig
Simple	Es darf nur eine einzelne Zeile als Stichwortangabe verwendet werden
Near	Es müssen genau zwei Stichwortzeilen angegeben werden
Date	Eine Stichwortangabe ist nicht erforderlich
Header	Es darf nur eine einzelne Zeile als Stichwortangabe verwendet werden
AND	Es müssen mindestens zwei Stichwortzeilen angegeben werden
OR	Es müssen mindestens zwei Stichwortzeilen angegeben werden

Wenn eine Einschränkung vorgibt, dass nur eine maximale Anzahl von Suchbegriffen angegeben werden darf, so erhält man keine Fehlermeldung, wenn doch mehr Begriffe angegeben werden. Das Programm verwendet allerdings nur die zuerst aufgeführten Begriffe.

5.4. Die Testkriterien

Für die in Punkt 3 angegebenen Tests wurde die folgenden Kriteriendatei verwendet.

```

CARACAS
ENTRY
TYP FileType
CAS N
VAL 1
txt

ENTRY
TYP Simple
CAS N
VAL 2
Call for paper

ENTRY
TYP Simple
CAS N
VAL 2
Calls for papers

```

```
ENTRY
TYP Simple
CAS N
VAL 2
CFPs
```

```
ENTRY
TYP Date
VAL 2
```

```
ENTRY
Typ Header
VAL 2
CFP
```

```
ENTRY
Typ Header
VAL 2
CALL FOR PAPER
```

5.5. *Beispiel für eine Konfigurationsdatei*

```
CARACAS

%Allgemeine Einstellungen
Triggervalue = 5
MaxLinkDepth = 4
StartTime = 00:00
StopTime = 23:59
TimeOut = 3
MailServer = smtp.web.de
MailUser = muehlich@web.de
MailLogin = muehlich
MailPWD = xxxxxxxxxxxxxx
MailSubject = Possible CFP by CARACAS-Webcrawler
Recipient = muehlich@in.tum.de
BlatLog = y

%Ab hier stehen die Start-URLs
BeginLinks
http://www.informatik.tu-muenchen.de/
http://dutetvg.et.tudelft.nl/~alex/CFP/
EndLinks

%Ab hier kommen die Kriterien
BeginCriterias
D:\Eigene Dateien\Studium\SEP\VB\TestCrit01.txt
EndCriterias
```