Universität Karlsruhe (TH)
Institut für Programmstrukturen und Datenorganisation
Prof. Dr. Walter F. Tichy

Verteilte Revisionskontrolle über das World Wide Web

Studienarbeit
07.06.1996

Bearbeiter: Jürgen Reuter
Betreuer: Prof. Dr. Walter F. Tichy
Dipl.-Inform. Stefan U. Hänßgen

Ich versichere, diese Arbeit selbständig verfaßt und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben.

Karlsruhe, den 7. Juni 1996 _____

(Jürgen Reuter)

Inhaltsverzeichnis

1 Einleitung und Problemstellung

Um bei der Erstellung umfangreicher Software den Überblick zu wahren, entsteht der Wunsch, über die Entwicklung Buch zu führen. Zur Automatisierung dieser Verwaltungsaufgabe bietet sich der Einsatz eines Software-Systems an. Derartige Systeme werden als *Revisionskontroll-* bzw. *Konfigurationsmanagementsysteme* (im folgenden kurz *RKS/KMS*) bezeichnet und werden seit vielen Jahren eingesetzt. Bekannte Beispiele hierfür sind etwa RCS [4] und SCCS [16].
Größere Software-Projekte werden heutzutage in der Regel nicht mehr von einzelnen Personen, sondern von Gruppen (*Teams*) erstellt. Daher kommt verstärkt der Wunsch nach der Möglichkeit einer räumlich verteilten, dezentralen Entwicklung von Software auf. Die fortschreitende weltweite Vernetzung von Rechnern geht dabei einher mit einem zunehmenden Bedarf an verteilten RKS/KMS. Das Internet und speziell das *World Wide Web* (*WWW*) beschleunigen diese Entwicklung zusätzlich.
Bis vor kurzer Zeit jedoch waren RKS/KMS ausschließlich als lokale Lösungen konzipiert. Zwar ist prinzipiell die dezentrale Nutzung eines lokalen Systems etwa via FTP und TELNET möglich; doch sind diese allgemeinen Basisdienste nicht speziell auf die Bedürfnisse ausgelegt, die mit dem Wunsch nach einem RKS/KMS einhergehen: im Falle eines lokalen RKS/KMS müßten die betreffenden Dateien etwa einzelnen von Hand via FTP übertragen oder in einem Verzeichnis gesammelt und anschließend gemeinsam übertragen werden; ferner müßten per TELNET zum richtigen Zeitpunkt die richtigen Befehle an das lokale RKS/KMS manuell übermittelt werden. Überdies erfordern die Basisdienste FTP und TELNET einen Benutzer-Account auf dem oder den jeweiligen Systemen, auf denen die Archive gehalten werden, was aus Sicherheitsgründen unerwünscht sein kann.
Die vorliegende Arbeit setzt sich daher zum Ziel, grundlegende Techniken zu untersuchen und zu implementieren, die zum Aufbau eines verteilten RKS/KMS erforderlich sind. Der Schwerpunkt liegt dabei nicht auf der einem RKS/KMS innewohnenden Funktionalität, sondern auf der technischen Realisierbarkeit der sich aus der Verteilung eines solchen Systems ergebenden Herausforderungen.
Die zunehmende Popularität und Verfügbarkeit des WWW legt es nahe, ein verteiltes RKS/KMS über einen zentralen WWW-Server zu realisieren, der mit dezentral verteilten Clients kommuniziert. Mit der dem WWW zugrunde liegenden Dokumentenbeschreibungssprache HTML [9] (*Hypertext Markup Language*), die auf dem Protokoll HTTP [8] (*Hypertext Transfer Protocol*) aufbaut, steht eine plattformunabhängige Kommunikationsschnittstelle zwischen Server und Client zur Verfügung. Unter Zuhilfenahme der zu ihrer Nutzung notwendigen, aber inzwischen weit verbreiteten Applikationen zur Anzeige von HTML-Dokumenten, sogenannten *Browsern*, und spezieller Server-Software steht damit eine Infrastruktur bereit, auf deren Basis sich komplexe Client/Server-Applikationen entwickeln lassen. Diese Infrastruktur rund um das WWW bietet somit alle erforderlichen Voraussetzungen zur Implementierung eines verteilten RKS/KMS.
In Anbetracht der geschilderten Situation konkretisiert sich das Ziel der vorliegenden Arbeit nunmehr zu der Idee, einen Prototypen für ein verteiltes RKS/KMS als Client/Server-Applikation unter Nutzung der Möglichkeiten des WWW zu entwerfen und zu implementieren.

2 Entwurf

Im Rahmen der vorliegenden Studienarbeit wurde unter der Bezeichnung *World Wide Revision Control* (kurz *WWRC*) ein Programm als Protoyp eines Systems zur verteilten Revisionskontrolle entworfen und implementiert. Der folgende Abschnitt untersucht die für den Entwurf des Prototypen relevanten Aspekte. Dabei gilt der Realisierung der Dialogschnittstelle und des Datentransportes über das WWW besonderes Augenmerk.

2.1 Versionskontrolle mit RCE

Bei der Implementierung von WWRC konnte auf bestehende Software zurückgegriffen und der Entwicklungsaufwand dadurch deutlich reduziert werden. So baut WWRC einerseits auf dem Revisionskontrollsystem RCE (*Revision Control Engine*) [5] auf.

Revisionskontrollsysteme wie RCS oder RCE speichern Revisionen einer Datei in einem Archiv ab, über das Buch geführt wird. Mit einer *Revision* einer Datei wird ein Zeitpunkt aus ihrer Entwicklungsgeschichte festgehalten. Eine als *Check-in* bezeichnete Operation dient dazu, eine neue Revision einer Datei in ihr zugehöriges Archiv aufzunehmen. Mit der als *Check-out* bezeichneten Operation wird eine bereits bestehende Revision aus einem Archiv ausgelesen oder eine neue Revision für ein nachfolgendes Check-in vorgemerkt. Der Vorgang des Check-in bzw. Check-out wird im folgenden auch kurz als *Einchecken* bzw. *Auschecken* bezeichnet.

Als ein Nachfolger von RCS bietet RCE eine Programmierschnittstelle (API) an, die ein direktes Aufsetzen von WWRC auf der von RCE zur Verfügung gestellten Infrastruktur ermöglicht. RCE bietet gegenüber RCS zudem einen deutlich verbesserten Delta-Algorithmus[1]. RCE selbst arbeitet dateiorientiert; es unterstützt also Revisionskontrolle, nicht aber Konfigurationsmanagement. Der Prototyp von WWRC beschränkt sich aus diesem Grunde ebenfalls auf Funktionen zur Revisionskontrolle, soll aber darüber hinaus die technischen Voraussetzungen für das Ein- und Auschecken vordefinierter Mengen von Dateien implementieren, ohne jedoch auf die Verwaltung dieser Mengen näher einzugehen.

2.2 Internet und WWW als Übertragungsmedium und Benutzerschnittstelle

Mit dem Ziel vor Augen, unter Nutzung des Internet das WWW als Benutzerschnittstelle zu verwenden, bietet sich die Implementierung von WWRC als WWW-Server an. Neben der geschilderten Einbindung von RCE wird somit auch die Nutzung bestehender Software für das WWW ermöglicht. Damit reduziert sich der Anteil an neu zu entwickelnder Software auf der Client-Seite erheblich. Für den Betrieb von WWRC sind dort neben einem marktüblichen Browser aus technischen Gründen lediglich zwei kleine zusätzliche Hilfsapplikationen erforderlich, deren Bedeutung in 2.4 näher erläutert wird. Die Interaktion mit dem Anwender läuft auf der Client-Seite vollständig über den Browser ab.

Abb. 1 verdeutlicht die Anordnung der einzelnen Bestandteile. Auf der einen Seite befindet sich ein Server, auf dem WWRC abläuft. WWRC hat direkten Lese- und Schreibzugriff auf die dort zentral gespeicherten Archiv-Dateien. Das Internet dient als ein Medium, über das die räumlich verteilten Benutzer von WWRC auf die Archiv-Dateien zugreifen können. Dazu wird zwischen dem WWRC-Server auf der einen Seite und dem Client als Rechner des Benutzers auf der anderen Seite eine TCP/IP-Verbindung oder, noch spezieller, eine WWW-Verbindung aufgebaut. Der dabei auf Client-Seite eingesetzte Browser zeigt zum einen die von WWRC erzeugten WWW-Seiten an. Zum anderen übermittelt er Daten an die erwähnten Hilfsapplikationen. Die Hilfsapplikationen ihrerseits bauen eine eigene TCP/IP-Verbindung zwecks Austausch weiterer Daten auf. Die Details hierzu werden in den Abschnitten 2.4 und 3.5 besprochen.

Bei dem vorgestellten Ansatz ergeben sich konzeptuelle Herausforderungen, die in den folgenden Unterabschnitten erläutert werden.

[1] Mittels eines *Delta-Algorithmus* kann ein Revisionskontrollsystem die – geeignet definierte – *Differenz* zweier Dateien berechnen. Speichert ein Revisionskontrollsystem diese Differenzen anstelle der vollständigen Revisionen, so führt dies in der Praxis zu einer meist deutlichen Datenreduktion (vgl. [2]) in den Archiven.

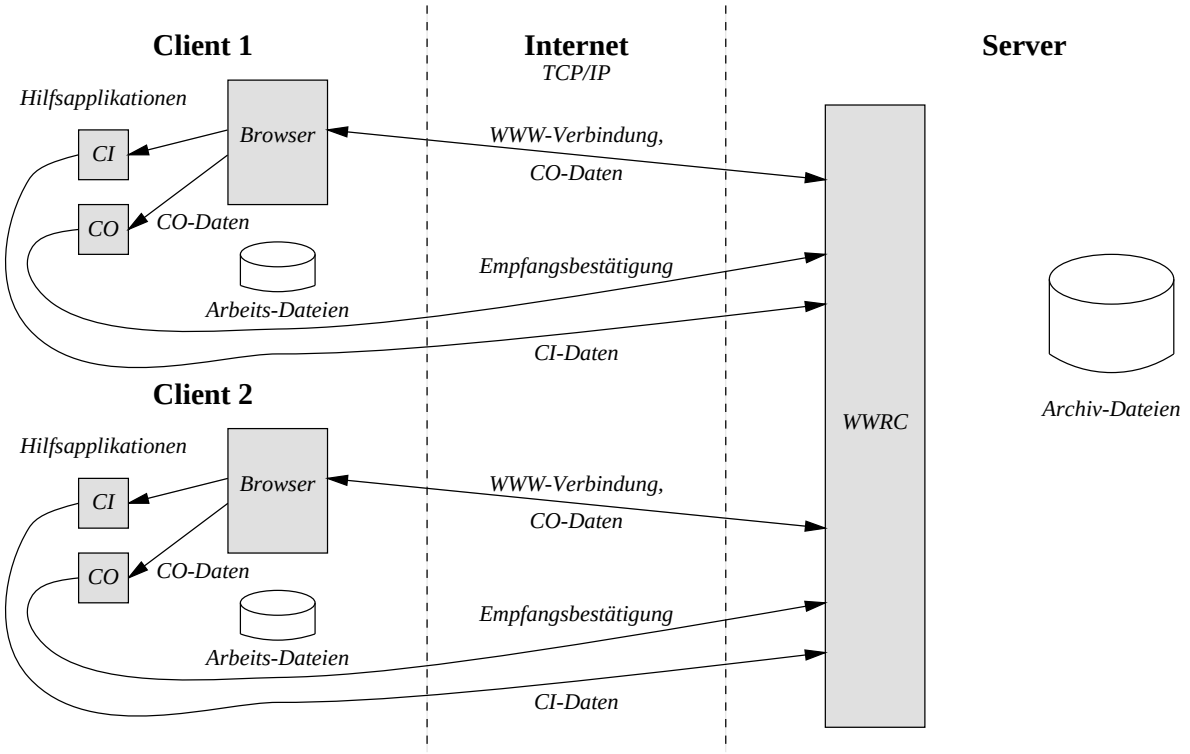

Abbildung 1: Übersicht

2.2.1 Übertragung von Daten im WWW

Eine grundlegende Funktion des WWW ist es, bei einem Server zentral gespeicherte Information einer Menge von räumlich verteilten Clients zur Verfügung zu stellen. Daher ist es ohne weiteres möglich, auch größere Datenvolumina vom Server zu einem Client zu senden. Bei einem System zur Revisionskontrolle wie WWRC soll der Transport großer Datenmengen jedoch in beiden Richtungen, also auch vom Client zum Server, möglich sein.
Zwar bietet HTTP mit den in HTML eingebundenen Sprachkonstrukten zur Übertragung von Formularen eine Möglichkeit, Daten vom Client an den Server zu schicken. Es gelten jedoch für die als Inhalt eines Formulars verwendbaren Zeichen diverse Einschränkungen, z.B. durch diverse HTML-inhärente Begrenzungszeichen, etwa zur Ende-Erkennung des Formularinhalts. Ferner müßten die zu übertragenden Daten, z.B. Dateien von der Festplatte des Client, erst einmal in ein solches Formular hineingelangen, was nicht ohne weiteres möglich ist.
Bei größeren Datenmengen besteht überdies der Wunsch nach einer weitreichenderen Kontrolle über den Datentransport. Hierzu gehört beispielsweise auch die Möglichkeit, nach einer durch Störung unterbrochenen Verbindung mit dem Daten-Transfer an einer geeigneten Stelle wiederaufsetzen zu können, ohne alle Daten nochmals neu zu übertragen.
Die unter der Bezeichnung *upload* bekannte Möglichkeit, Daten vom Client zum Server zu schicken, ist nicht standardisiert und wird nur von manchen Browsern unterstützt.
Es entsteht somit der Wunsch nach einer alternativen Methode der Datenübertragung vom Client zum Server, wenn man eine Unabhängigkeit von spezifischen Browsern erreichen will. Einen Ausweg bietet der in 2.4 beschriebene Einsatz von Hilfsapplikationen[2].

[2]Als *Hilfsapplikation* (*Helper Application*) oder auch *External Viewer* werden im Umfeld des WWW externe

Im umgekehrten Fall der Übertragung von Dateien vom Server zum Client sollen diese Daten möglichst automatisch beim Client abgelegt werden, ohne daß der Benutzer manuell ein Zielverzeichnis angeben müßte. Damit soll neben der Erzielung höheren Komforts auch die Anzahl an Fehlermöglichkeiten durch falsche Eingaben des Benutzers verringert werden. WWRC bedient sich hier ebenfalls einer Hilfsapplikation.

2.2.2 Zustandslosigkeit im WWW

Eine grundlegende Philosophie des WWW ist die Zustandlosigkeit des Servers. Hierunter ist zu verstehen, daß eine wiederholte Anfrage an den Server wieder das gleiche Dokument liefert. Der Inhalt einer WWW-Seite ist damit kontextunabhängig und nur durch die aktuelle Anfrage bestimmt. Dieses Konzept steht leider in diametralem Gegensatz zu den Erfordernissen eines sitzungsorientierten, kontextabhängigen Systems wie WWRC. Der Benutzer von WWRC soll Möglichkeiten zu Änderungen eines gemeinsam verwalteten Datenbestandes erhalten. Dieser Datenbestand stellt bereits einen Zustand dar. Hinzu kommen noch etwaige benutzerspezifische Daten. In jedem Fall muß die Zustandslosigkeit auf die eine oder andere Weise durch Speicherung zusätzlicher Informationen durchbrochen werden. Dies kann zu Inkonsistenzen führen, deren Ursachen und Möglichkeiten zu ihrer Vermeidung in 2.3 näher aufgezeigt werden.

2.2.3 Server-Funktionalität

Mit dem W3C-Server [6], vormals bekannt als CERN-Server, und dem NSCA-Server [7] existieren zwei weit verbreitete WWW-Server, deren Quellcode frei erhältlich ist. Beide Server gestatten überdies das Einbinden von ausführbaren Programmen (*CGI-Skripte*[3]). Damit bietet sich auf den ersten Blick eine Realisierung von WWRC als CGI-Skript unter Verwendung eines dieser Server an. Bei näherer Betrachtung scheidet diese Möglichkeit jedoch aus: wegen der Größe der Applikation WWRC würde das ständige Starten des CGI-Skriptes durch den Server unnötig Ressourcen verbrauchen und zu langen Antwortzeiten führen.
Denkbar wäre weiterhin die Realisierung von WWRC als ständig im Hintergrund laufender Prozeß (*Dämon*); ein kleines CGI-Skript genügte dann, um ankommende Anfragen vom Server an den WWRC-Dämon mittels Interprozeßkommunikation weiterzuleiten. Der WWRC-Dämon könnte dann seine WWW-Seite dynamisch erzeugen und über das CGI-Skript zurück an den Server zur Weitergabe in das Netz übermitteln. Bei einer solchen Vorgehensweise würde die Rolle des Servers allerdings im wesentlichen auf das Entgegennehmen der Anfrage und das Zurückschicken der von WWRC generierten WWW-Seite degradiert. Diese Aufgabe mit in WWRC zu integrieren, stellt jedoch kein größeres Problem dar und vermeidet überdies den Umweg der Interprozeßkommunikation.
Eine Implementierung von WWRC als eigenständiger Server ist zudem wesentlich flexibler: durch die vollständige Kontrolle über die in HTTP/1.0 definierten Anfragefelder (*request fields*) und Antwortfelder (*response fields*) einschließlich des Server-Status vermag ein eigenständiger WWRC-Server zusätzliche Informationen vom Client auszuwerten, beliebige Authentifikations- und Autorisierungsverfahren zu implementieren und beispielsweise auch die aus mehreren Dokumenten zusammengestzten sogenannten *Multipart-Messages* zu erzeugen.

Programme bezeichnet, die vom Browser aus gestartet werden. Sie dienten ursprünglich dazu, Dokumente anzuzeigen, für deren Format der Browser keine interne Darstellungsmöglichkeit eingebaut hat. So wird unter dem Betriebssystem Unix beispielsweise häufig die Applikation `ghostview` vom Browser aus aufgerufen, um Dateien anzuzeigen, die im PostScript-Format vorliegen.

[3] Mit *CGI* (*Common Gate Interface*, [11]) bezeichnet man eine normierte Schnittstelle für Server-Software zur Einbindung externer ausführbarer Programme (*CGI-Skripte*). Mit ihrer Hilfe ist es möglich, die Funktionalität des Servers insbesondere auch um die dynamische Erzeugung von Dokumenten zu erweitern.

2.3 Zugriffskontrolle und Konsistenzprüfung

WWW-Seiten sind i.a. für alle an das Internet angeschlossenen Teilnehmer zugänglich. Dies spiegelt den ursprünglichen Charakter des WWW wider, Information für jeden Teilnehmer frei zur Verfügung zu stellen. Soll – wie bei WWRC wünschenswert – die Menge zugangsberechtigter Teilnehmer eingegrenzt werden, so benötigt man einen Mechanismus zur *Zugriffskontrolle*. Eine Zugriffskontrolle läßt sich einerseits durch Überprüfen der Internet-Adresse desjenigen Rechners, der eine Anfrage an den Server richtet, bewerkstelligen. Andererseits ist es möglich, jeder Anfrage eine Benutzerkennung und/oder Paßwort zur Autorisierung hinzuzufügen. Bei falscher oder fehlender Autorisierung liefert der Server dann das angeforderte Dokument nicht aus.
Mit dem fortschreitenden Wandel des WWW vom Angebot rein statischer Information hin zu interakiven Dialogen entsteht die Notwendigkeit, Daten benutzer- und kontextabhängig zu behandeln. Dazu ist auf der einen Seite eine Benutzererkennung erforderlich, die jedoch bereits im Rahmen der Zugriffskontrolle mit realisiert werden kann. Auf der anderen Seite muß i.a. eine Konsistenzprüfung erfolgen, da WWRC, wie in 2.2.2 beschrieben, als sitzungsorientierte Applikation nicht zustandsfrei ist; alte WWW-Seiten können Elemente enthalten, die nach dem aktuellen Stand der Sitzung nicht mehr gültig sind. Die Fortsetzung des Dialoges von einer alten WWW-Seite aus, deren Inhalt nicht mehr den aktuellen Zustand der Datenbasis wiedergibt, muß daher berücksichtigt werden. Die Gültigkeit einer Anfrage an den Server ist also i.a. kontextabhängig, und es muß somit in jedem Falle eine Prüfung erfolgen, ob die Anfrage an den Server im Kontext des vorausgegangenen Dialoges wunschgemäß bearbeitet werden kann oder abgewiesen werden muß.
In einem ersten Entwurfsansatz der Zugriffskontrolle und Konsistenzprüfung wurden diese beiden Aspekte verbunden: Nach Prüfung der Zugangsberechtigung zu einer Einstiegsseite über eine Benutzername- und Paßwortabfrage wird mit jeder Seite ein Zeitstempel übertragen, der den Zeitpunkt der Generierung der WWW-Seite markiert. Dieser Zeitstempel dient beim Anfordern einer neuen Seite als (für den Benutzer nicht explizit einzugebendes) neues Paßwort. Diese Vorgehensweise kommt einem ständig wechselnden Paßwort nahe; das eigentliche, vom Benutzer einzugebende Paßwort wird nur jeweils ein einziges Mal zu Beginn einer Sitzung eingegeben und über das Netz übertragen; Lauschangriffe werden dadurch erschwert. Zugleich wird damit auch die Möglichkeit unterbunden, mit der „Back“-Taste des Browsers eine alte WWW-Seite anzuzeigen und von dort aus den Dialog fortzusetzen.
Nachteilig ist bei diesem Verfahren jedoch, daß die Fortsetzung des Dialoges von einer veralteten WWW-Seite aus wegen ihres dann ungültigen Zeitstempels grundsätzlich nicht mehr möglich ist, selbst wenn dies zu keiner Konsistenzverletzung führte. Ferner mißfällt aus Gründen der Flexibilität die Festlegung auf ein spezielles Autorisierungsverfahren. Daher wurde in einer späteren Phase des Projekts die Autorisierung von dem Zeitstempel-Mechanismus entkoppelt. WWRC implementiert jetzt das *Basic Protection Scheme* gemäß [8], zu dem im Abschnitt 3.4 nähere Erläuterungen folgen. Es besteht damit nun die Möglichkeit, die Granularität des Zeitstempel-Mechanismus zur Konsistenzprüfung beliebig zu verfeinern.

2.4 Check-in, Check-out

Entscheidend für eine einfach und effizient nutzbare Oberfläche ist eine weitgehende Automatisierung grundlegender Operationen wie dem Check-in und dem Check-out (vgl. 2.1), im folgenden auch kurz mit *CI* und *CO* bezeichnet. Neben dem Umgang mit einzelnen Dateien interessiert insbesondere auch die Anwendung der CI/CO-Operationen auf eine Gruppe von Dateien (im folgenden kurz *Multi-CI/CO*). Die organisatorischen Aspekte wie die – vor allem bei Multi-CI/CO – notwendige Selektion der gewünschten Revisionen bleibt Aufgabe einer Konfigurationsverwaltung, die nicht unmittelbar Gegenstand der vorliegenden Arbeit ist (vgl. Abschnitt 6). WWRC

verwaltet zur Realisierung des Multi-CO lediglich eine Liste, die *Pickup-Liste*, in der sich hinter jedem Eintrag eine Menge von Dateien samt Revisionsbezeichnung verbirgt. Eine detaillierte Spezifikation und Implementierung von Operationen auf dieser Liste bleibt einer zukünftigen Arbeit überlassen. Es interessiert hier vielmehr der Aspekt der Kommunikation zwischen Client und Server zum Ziele der Übertragung von Daten, insbesondere auch größeren Dateien. Ziel ist dabei das Anstoßen einer solchen Kommunikation durch Aktivieren eines einzelnen Hyperlink[4].
In einem ersten Entwurfsansatz liegt die Idee nahe, die Übertragung von Dokumenten vom Server zum Client durch einfaches, direktes Herunterladen der entsprechenden Dokumente zu realisieren, wie man dies von zahlreichen WWW-Seiten her kennt. Der Benutzer klickt dabei auf einen Hyperlink, mittels dessen dem Server Name und Ort der entsprechenden Datei mitteilt werden; der Server übermittelt dann in seiner Funktion als Datei-Server das angeforderte Dokument. Diese grundlegende Funktion beherrschen alle gängigen WWW-Server; WWRC als eigenständige Applikation wäre hierfür allein überflüssig.
Mit dieser Vorgehensweise läßt sich aber keine Revisionskontrolle, geschweige denn Konfigurationskontrolle, bewerkstelligen. So entziehen sich hierbei die Dokumente jedweder Kontrolle durch den Server wie den Client. Der Server muß aber – zumindest, wenn das gesamte Check-out in einem einzigen Schritt vollzogen werden soll – die betreffende Datei zunächst auschecken und danach zum Client übertragen. Im Fall eines Multi-CO müssen mehrere Dateien beim Server zusammengefaßt und beim Client wieder voneinander getrennt werden; hinzu kommen eventuell noch eine spezielle transportgerechte Codierung oder Verschlüsselung der Dateien; bei größen Dateien kann es sich überdies lohnen, diese vor der Übertragung zu komprimieren und beim Client wieder zu dekomprimieren.
Auch das Verhalten des Browsers bei einem durch direktes Herunterladen von Dateien realisierten Check-Out genügt nicht den Ansprüchen an eine Revisionsverwaltung. Da nämlich CI/CO-Operationen i.a. auf beliebigen, dem Browser möglicherweise unbekannten Dokumententypen erfolgen, muß man beim direkten Herunterladen damit rechnen, daß der Browser den Benutzer entscheiden läßt, wo das betreffende Dokument abgelegt werden soll. Hier wäre eine Automatisierung des Abspeichervorganges wünschenswert.
Das Verhalten des Browsers kann überdies zu gänzlich ungewolltem Verhalten führen, etwa wenn beim Auschecken einer HTML-Datei oder von Bilddaten die betreffende Datei nicht zur Ablage auf Festplatte bereitgestellt, sondern vom Browser sogleich angezeigt wird. Dann muß der Benutzer selbst das betreffende Dokument explizit sichern.
All diese Herausforderungen lassen sich mit dem Konzept des direkten Herunterladens nur unzureichend oder gar nicht umsetzen. Unter Verwendung spezieller Hilfsapplikationen (vgl. 2.2.1) können die Operationen Check-in und Check-out jedoch wie folgt realisiert werden:
Beim Check-in schickt der Client eine Anfrage in einer speziellen Syntax an den Server. Der Server erkennt das Ansinnen des Client, führt lokal das Check-out durch, bereitet ggf. die Datei oder die Dateien für die Übertragung vor und führt die Übertragung durch. Den hierbei übermittelten Daten wird ein spezieller Dokumententyp aufgeprägt, der den Browser veranlaßt, eine spezielle Hilfsapplikation auf dem Client ablaufen zu lassen. Diese Hilfsapplikation nimmt die Daten entgegen, entpackt sie gegebenenfalls und speichert sie automatisch ab.
Auch im Falle des Check-in schickt der Client eine Anfrage an den Server. Der Server teilt dem Client als Antwort die näheren Modalitäten der folgenden Übertragung mit. Diese Daten werden wie beim Check-out an eine Hilfsapplikation weitergeleitet, die dann die benötigten Dateien direkt an den Server schickt.
Die technischen Details des hier nur kurz skizzierten Ablaufs des Check-in und des Check-out werden in 3.5 näher erläutert.

[4]Unter einem *Hyperlink* versteht man einen symbolischen Bezeichner, hinter dem sich eine Anfrage an einen WWW-Server verbirgt. Die Anfrage wird durch Anklicken des Hyperlink ausgelöst.

3 Implementierung

Die Implementierung von WWRC erfolgte in der Programmiersprache Modula-3 [3] auf einer Sun Sparc Workstation unter dem Betriebssystem SunOS 4.1.3. Der folgende Abschnitt zeigt beispielhaft einige Aspekte auf, denen bei der Umsetzung des Entwurfs in ein ablauffähiges Programm besondere Bedeutung zukam.

3.1 Ablaufform

Die derzeitige Implementierung von WWRC vermeidet bislang direkte Socketprogrammierung; stattdessen erfolgt die Kommunikation zwischen WWRC und TCP/IP aus Gründen der Einfachheit über den Dämon `inetd`. `inetd` ist ein unter Unix übliches Werkzeug, das als ständig laufender Hintergrundprozeß über das Netz hereinkommende Daten entgegennimmt und anschließend die gewünschte Applikation – hier WWRC – startet. WWRC stehen die Daten über das Standardeingabegerät zur Verfügung; Ausgaben auf das Standardausgabegerät werden von `inetd` an den Clienten weitergeleitet.
Nachteilig ist bei dieser Vorgehensweise jedoch, daß bei jeder Anfrage WWRC in jeweils einer eigenen Instanz neu gestartet wird. Dies führt zu einem eigentlich vermeidbarem Verbrauch von Ressourcen und einer unnötig langen Antwortzeit des Servers. Wünschenswert wäre daher eine Realisierung von WWRC als Dämon. Die Ladezeiten von WWRC entfielen, die Antwortzeit verringerte sich deutlich.
Im Prinzip gestattet die prozedurale Ablaufform von WWRC ohne größere Umstrukturierungen eine Umstellung auf den Betrieb als Dämon. Dazu muß lediglich ein Sekretär als Endlosschleife implementiert werden, der mit jeder hereinkommenden Anfrage einen eigenen Thread als Aktivitätsbahn startet, auf der die Anfrage wie bisher prozedural abgearbeitet wird. Zusätzlich müssen ggf. noch Maßnahmen zur Behandlung konkurrierender Lese-/Schreibzugriffe auf die gemeinsame Datenbasis durch die nunmehr i.a. parallel ablaufenden Aktivitäten ergriffen werden.

3.2 HTTP/0.9 versus HTTP/1.0

Der Einsatz von Hilfsapplikationen erfordert die Vereinbarung eines speziellen Dokumententyps. Das Erkennen des Dokumententyps erfolgt unter HTTP/0.9 an der Endung der URL. Damit muß auf einer WWW-Seite bereits der Typ des nächsten zu übertragenden Dokumentes feststehen, der sich hinter einem Hyperlink verbirgt. Unter HTTP/1.0 kann der Server den Dokumententyp in dem im Anschluß an den Server-Status folgenden *Response Header* im Feld *Content-Type* übermitteln; damit kann die Entscheidung über den zu übertragenden Dokumententyp bis zur Übertragung selbst hinausgezögert werden. Dies ist vor allem dann interessant, wenn der Aufruf einer Hilfsapplikation beabsichtigt war, aufgrund eines Fehlers aber dann die Übertragung des betreffenden Dokumentes doch nicht erfolgen kann. Der Server kann dann stattdessen eine Fehlermeldung als HTML-Seite an den Client schicken; der Browser erkennt am Response Header, daß es sich um eine HTML-Seite erkennt und ruft daher keine Hilfsapplikation auf, sondern stellt die Fehlermeldung als HTML-Seite dar.
Ein zweiter Grund für den vorzugsweisen Einsatz von HTTP/1.0 gegenüber HTTP/0.9 ist die Möglichkeit, *Multipart Messages* zu übermitteln. In HTTP/0.9 kann durch Anklicken eines Hyperlink nur jeweils ein einzelnes Dokument übertragen werden. Dadurch ist es möglich, entweder eine Hilfsapplikation zu starten oder eine neue HTML-Seite zur Anzeige bringen zu lassen, jedoch nicht beides hintereinander in einem Zuge. Wie in 2.4 erläutert, sollen zur Ausführung von CI/CO-Operationen Hilfsapplikationen aufgerufen werden. Da derartige Operationen aber auch semantische Änderungen nach sich ziehen, ist es wünschenswert, diese Änderungen auch

visuell durch Anzeige einer neuen HTML-Seite anzuzeigen. Hierfür benötigt man einen Mechanismus, der es gestattet, mehrere Dokumente zusammengefaßt an den Client zu übermitteln. Unter HTTP/1.0 ist dies mit Dokumenten des Typs *multipart/mixed* möglich.
WWRC verwendet aus diesen Gründen HTTP/1.0 zur Implementierung der CI/CO-Operationen. In der Funktion als reiner Datei-Server antwortet WWRC gemäß HTTP/0.9, wenn der Client eine dementsprechende Anfrage stellt, um auch ältere Browser zumindest in diesem Punkte zu unterstützen. Mit der fortschreitenden Verbreitung der moderner Browser spielt HTTP/0.9 aber eine stetig unbedeutendere Rolle.

3.3 Zeitstempel-Mechanismus zur Konsistenzprüfung

Wie in 2.3 angedeutet, verwendet WWRC einen Zeitstempel-Mechanismus zur Konsistenzprüfung eingehender Anfragen an den Server. Dabei wird mit jeder WWW-Seite ein Zeitstempel übermittelt, der Datum und Uhrzeit der Generierung wiedergibt. Technisch ist dies realisierbar, indem man in jede URL[5] einer generierten WWW-Seite entsprechende Information hineincodiert. Erfolgt eine Anfrage mit ungültigem Zeitstempel an den Server, so erkennt der Server daran eine potentielle Konsistenzverletzung und kann entsprechend reagieren. In der aktuellen Implementierung wird der Dialog in solch einem Falle zu einer unkritischen Stelle (dem *Hauptmenü*, vgl. Abb. 6) zurückgesetzt und eine entsprechende Warnung mit ausgegeben. Der so implementierte Zeitstempel-Mechanismus garantiert die geforderte Konsistenz des Dialoges – zumindest solange keine vorsätzliche Manipulation an einer URL erfolgt. Eine solche Manipulation kann schlimmstenfalls zur Meldung eines internen Fehlers führen, ohne jedoch den Bestand der Datenbasis zu gefährden.

3.4 Basic Protection Scheme als Zugriffskontrolle

Dieses Autorisierungsverfahren sieht vor, daß der Server bei einer Anfrage dem Client anstelle des angeforderten Dokumentes eine Aufforderung zur Autorisierung zurückschicken kann. Diese Antwort des Servers enthält nähere Informationen über das zu verwendende Autorisierungsverfahren. Im Falle des Basic Protection Scheme kann der Browser nun die Anfrage an den Server wiederholen. Dabei übermittelt er in einem speziellen Anfragefeld das – gegebenenfalls vom Benutzer einzugebende – Benutzerkennwort samt Paßwort. Die Übertragung von Benutzername und Paßwort erfolgen unverschlüsselt; das Basic Protection Scheme ist daher nicht sicher gegen Lauschangriffe. Zusätzliche Schutzmaßnahmen lassen sich aber problemlos hinzufügen (vgl. 6).

3.5 Check-in, Check-out

Die Operationen Check-in und Check-out bedienen sich in der gegenwärtigen Implementation von WWRC eines Modells gespiegelter Verzeichnisstrukturen (vgl. Abb. 2). Danach befinden sich alle Archive beim Server in einer Verzeichnisstruktur unterhalb eines ausgezeichneten Wurzelverzeichnisses `SERVER_ROOT`. Beim Client findet sich ebenfalls ein ausgezeichnetes Wurzelverzeichnis `CLIENT_ROOT`, unterhalb dessen sich die Arbeitsdateien befinden. WWRC sorgt – soweit möglich – nun dafür, daß der Verzeichnispfad jeder Arbeitsdatei relativ zu `CLIENT_ROOT` übereinstimmt mit dem Verzeichnispfad der zugehörigen Archivdatei relativ zu `SERVER_ROOT`.
Die folgenden beiden Unterabschnitte erläutern die in 2.4 kurz skizzierten Abläufe in ihren technischen Details. Wir betrachten zunächst den Vorgang des Check-out, da sich dieser Mechanismus einfacher gestaltet.

[5]Mit *URL* (*Uniform Resource Locator*, vgl. [14]) bezeichnet man eine weltweit eindeutige Adresse im WWW, unter der ein Dokument aufgefunden werden kann.

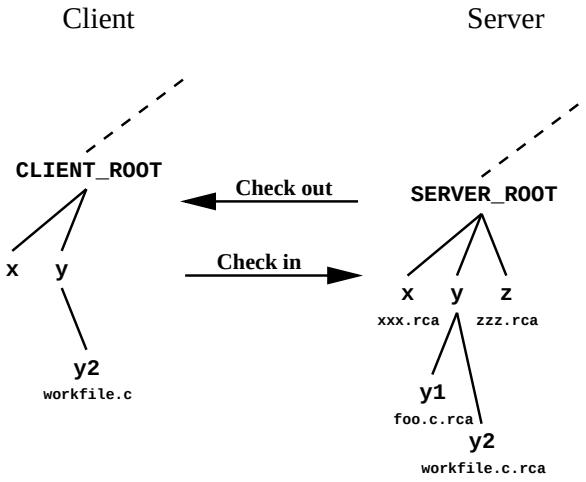

Abbildung 2: gespiegelte Verzeichnisstruktur beim Client – hier als echte Teilmenge der Verzeichnisstruktur des Servers

3.5.1 Check-out

Abbildung 3 illustriert den im folgenden beschriebenen Vorgang. Aktiviert der Anwender auf einer WWW-Seite von WWRC eine Schaltfläche mit der Aufschrift „Check out“, so sendet der Browser eine entsprechende Anfrage an den Server. Der Server entnimmt der Anfrage, beim Multi-CO unter Berücksichtigung seiner eigenen Datenbasis, welche Datei bzw. welche Dateien ausgecheckt werden sollen. Er führt zunächst ein lokales Check-out durch, faßt im Falle eines Multi-CO die hierbei entstandenen Dateien zusammen, komprimiert diese, codiert diese in eine übermittlungsgerechte Form und sendet diese schließlich an den Client. Den Daten wird gemäß HTTP/1.0 ein Response Header vorausgeschickt, dem zur Identifikation des Dokumententyps eine spezielle Kennung hinzugefügt wird, die zur Ausführung einer speziellen Hilfsapplikation führt. Diese kleine Hilfsapplikation ist im Falle von WWRC in der Skriptsprache *Perl*[6] realisiert. Das Perl-Skript nimmt die Daten entgegen, decodiert und dekomprimiert sie, spaltet sie ggf. in einzelne Dateien auf und legt sie in dem Verzeichnis ab, das zusammen mit der Datei übermittelt wird. Anschließend sendet das Skript an den Server – syntaktisch gesehen in Form einer Anfrage – eine Bestätigung für den Erhalt der Daten. Der Server kann ein etwaiges Fehlen dieser Bestätigung bei Erzeugen der nächsten WWW-Seite berücksichtigen.
Soll eine neue Revision einer einzelnen Datei ausgecheckt werden, so wird zuvor eine WWW-Seite generiert, die ein Formular enthält, in das man zusätzliche Angaben eintragen kann; dazu zählen insbesondere der Revisionsname, der Status der Revision sowie ein Kommentar.

3.5.2 Check-in

Beim Check-in besteht die Aufgabe, durch das Anklicken eines Hyperlink die Übertragung von Daten vom Client zum Server auszulösen. Abbildung 4 illustriert den im folgenden beschriebenen Vorgang. Aktiviert der Anwender auf einer WWW-Seite von WWRC eine Schaltfläche mit der Aufschrift „Check in“, so wird durch eine entsprechende Anfrage an den Server der Vorgang des Check-in eingeleitet.

[6]Perl genießt den Vorteil, auf vielen verschiedenen Plattformen verfügbar zu sein; die Portabilität der Hilfsapplikation wird dadurch erhöht.

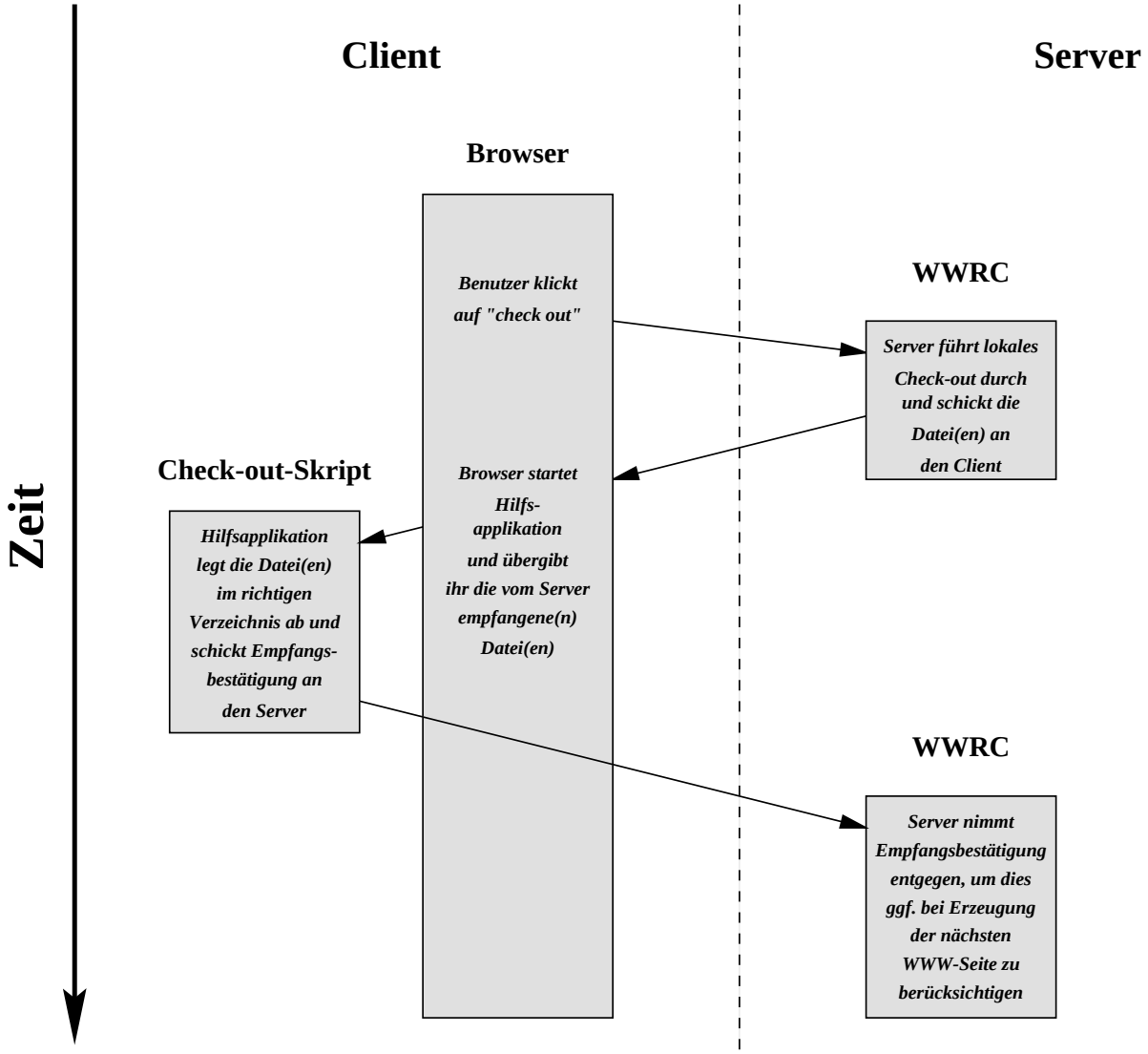

Abbildung 3: Interner Ablauf eines Check-out

Wie in 3.2 angedeutet, ermöglichen es die Multipart Messages unter HTTP/1.0 dem Server, mehrere Dokumente auf eine einzige Anfrage hin zu verschicken. Im Falle des Check-in werden zwei Dokumente an den Client gesendet.
Das erste Dokument enthält Informationen darüber, welche Dateien eingecheckt werden sollen, damit der Client diese Dateien an den Server schickt. Diese beim Server gespeicherte Information ergibt sich aus dem Kontext des vorausgegangenen Dialoges.
Für das erste Dokument ist wiederum ein eigener Dokumententyp definiert, den der Browser dem Response Header aus der Antwort des Servers entnimmt. Dieser Dokumententyp ist ebenfalls einer speziellen Hilfsapplikation zugeordnet, die der Browser nach Empfang des Dokumentes ausführen läßt. Die Hilfsapplikation ist abermals durch ein kleines Perl-Skript realisiert.
Das zweite Dokument wird nicht sofort vom Server abgeschickt; daher bleibt die Verbindung zwischen Server und Browser während der Ausführung des Perl-Skripts bestehen.
Das Perl-Skript baut kurzzeitig eine zweite, eigene Verbindung zum Server auf. Über diese Verbindung sendet das Perl-Skript die angeforderten Daten an den Server. Der Server führt nun das Check-in aus.
Erst jetzt sendet der Server das zweite Dokument über die immer noch bestehende Verbindung zum Browser: er übermittelt als zweites Dokument eine neue HTML-Seite, aus der das erfolgreich abgeschlossene Check-in ersichtlich wird. Erhält der Server bis zum Ablauf einer vorgegebenen Frist keine Daten vom Client, so wird stattdessen eine HTML-Seite mit einer entsprechenden Fehlermeldung generiert und an den Browser geschickt.
Nach dem Erhalt dieses zweiten Dokumentes ist die Übermittlung beendet, und die Verbindung wird abgebaut.
In der vorliegenden Implementierung läuft der Server noch nicht als Dämon; daher verbirgt sich hinter jeder Verbindung zwischen Server und Client eine eigene Programm-Instanz; die Kommunikation zwischen den Instanzen erfolgt derzeit noch über das Datei-System. Diese implementierungsspezifischen Details wurden in Abb. 4 aus Gründen der Übersichtlichkeit weggelassen.
Bei Umstellung der Ablaufform von WWRC auf Dämonbetrieb kann man zu einer Kommunikation übergehen, die innerhalb der dann einzigen Serverinstanz zwischen ihren einzelnen Threads abläuft. Interthreadkommunikation innerhalb eines gemeinsamen Adreßraums ist deutlich einfacher und schneller als die Kommunikation zwischen verschiedenen Prozessen. Denkbar wäre auch eine rein sequentielle Lösung innerhalb eines einzelnen Thread, der zunächst das erste Dokument erzeugt, auf Antwort vom Client über eine zweite TCP/IP-Verbindung wartet und schließlich das zweite Dokument übermittelt.

4 Ein Beispiel-Dialog

Um einen Einblick zu erhalten, wie sich WWRC dem Benutzer gegenüber präsentiert, soll hier exemplarisch ein kurzer Beispiel-Dialog betrachtet werden.
Die von WWRC generierten Seiten erfordern, wie in 2.3 geschildert, die Autorisierung durch den Benutzer. WWRC verwendet das Basic Protection Scheme; daher fordert der Browser zu Beginn einer Sitzung den Benutzer auf, Benutzerkennung und Paßwort einzugeben, sofern dies nicht bereits erfolgt ist (vgl. Abb. 5).
Sodann wird vom Server eine WWW-Seite generiert, die dem Benutzer verschiedene mögliche Aktionen zur Auswahl stellt (vgl. Abb. 6). Es stehen die für ein Revisionskontrollsystem typischen Funktionen wie die Anzeige des Revisionsgraphen eines Archivs einschließlich der darauf anwendbaren Operationen Check-in und Check-out (über den Menüpunkt „Revisionsgraph“) zu Verfügung; auch ein Multi-CO ist wie in 3.5.1 beschrieben implementiert. Man kann sich ferner das von RCE verwaltete Logbuch eines Archivs anzeigen lassen. Die Funktionen zum Editieren der für das Multi-CI/CO relevanten Pickup-Liste (vgl. 2.4) einschließlich der Dateiauswahlbox

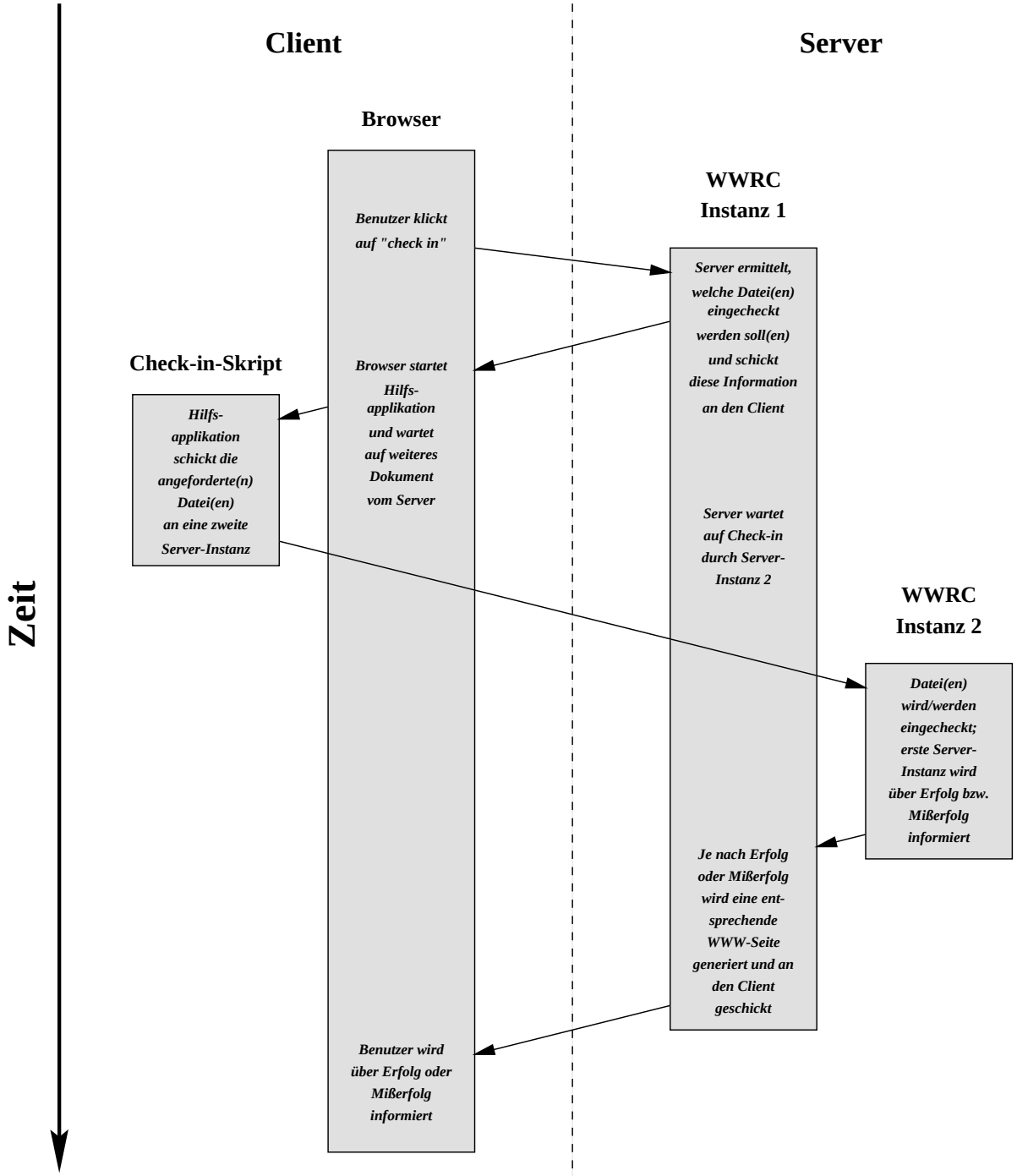

Abbildung 4: Interner Ablauf eines Check-in

(„File Browser“) und der Funktion zum Anlegen eines neuen Archivs sind eng verbunden mit dem zu wählenden Konzept für ein Konfigurationsmanagement, was nicht primäre Aufgabe dieses Prototyps ist. Daher sind diese Funktionen nicht bzw. nur teilweise implementiert; ihre nähere Behandlung ist Gegenstand zukünftiger Arbeit (vgl. Abschnitt 6). Schließlich bietet sich noch die Möglichkeit, das Paßwort zu ändern, das in der WWRC eigenen Datenbasis abgelegt und somit unabhängig von etwaigen Paßwörtern des Server-Betriebssystems verwaltet wird.
Nach Anklicken des Menüpunktes „Revisionsgraph“ wird der hier vollständige Revisionsgraph einer Datei angezeigt, wie Abb. 7 veranschaulicht. Der Übersicht halber lassen sich Teilbäume ausblenden, wenn man auf den entsprechenden Knoten mit der Aufschrift „hide“ klickt. Ausgeblendete Teilbäume werden durch einen Hyperlink mit der Bezeichnung „show“ gekennzeichnet, bei dessen Aktivierung der entsprechende Teilbaum wieder sichtbar wird. Durch Anklicken einer Revision wird diese selektiert, und es erscheinen nähere Informationen zu dieser Revision, darunter Zeitstempel, Autor, Revisionsstatus und Kommentar zur selektierten Revision, sofern verfügbar. Der Revisionsgraph könnte im Prinzip auch als vom Server automatisch generierte Grafik (als *clickable image*) übermittelt werden; die hier gewählte rein textuelle, vorformatierte Version führt indes zu deutlich geringerem Übertragungsvolumen.
Durch Anklicken der Schaltfläche „Back to Menu“ gelangt man zurück zum Hauptmenü (Abb. 6). In Abb. 8 erkennt man, wie von der Pickup-Liste ein Eintrag selektiert wurde. Dieser *Alias*[7] steht stellvertretend für eine Menge von Dateien. Durch Aktivieren der Funktion „Multi CO“ werden diese Dateien gemäß dem in 3.5.1 beschriebenen Verfahren ausgecheckt. Abb. 9 zeigt die Ausgaben des derweil beim Client ablaufenden Perl-Skripts auf das Standardausgabegerät bzw. Standardfehlerausgabegerät. Diese Ausgaben werden vom Browser in einem eigens hierfür geöffneten Fenster als Kontrollinformation für den Benutzer dargestellt. Man erkennt in diesem Beispiel eine Liste der übermittelten Dateien sowie den Namen und Portadresse des Servers. Das Skript meldet schließlich das erfolgreiche Absenden der Empfangsbestätigung an den Server.

5 Vergleich mit anderen Systemen

Neben WWRC verfolgen zwei andere dem Autor bekannte Projekte das Ziel eines Systems zur verteilten Revisions- oder Konfigurationskontrolle über das WWW. Diese sollen in den folgenden beiden Unterabschnitten im Vergleich zu WWRC kurz betrachtet werden.

5.1 Integrity Engine

Von Mortice Kern Systems (MKS) Inc. wird unter der Bezeichnung *Integrity Engine* ein Autorensystem für die verteilte Entwicklung von Projekten angeboten. Einer Kurzbeschreibung dieses Produktes (vgl. [17]) zufolge ist dieses System speziell auf einen Browser (`Netscape`) zugeschnitten; mit anderen Browsern kann nur eine eingeschränkte Funktionalität genutzt werden. Es wird insbesondere nahegelegt, einen Browser zu verwenden, der einen eingebauten Texteditor zur Verfügung stellt, um ein editiertes Dokument sogleich an den Server zurückzuschicken. Das Dokument bleibt dadurch die gesamte Zeit unter der Kontrolle des Browsers. Die Integrity Engine verwendet überdies eine erweiterte Syntax für URLs, deren Benutzung von dem verwendeten Browser unterstützt werden muß. Ein Umstieg auf einen anderen Browser ist daher i.a. nicht ohne weiteres möglich.
Auch WWRC definiert eine spezielle Syntax für URLs, bleibt dabei allerdings im durch `RFC1738` [14] gesteckten Rahmen. Eine Anpassung des Browsers ist daher nicht notwendig. WWRC

[7] Mit *Alias* bezeichnet man einen leicht zu merkenden symbolischen Bezeichner, der stellvertretend für einen komplexeren und oftmals wesentlich längeren Ausdruck steht.

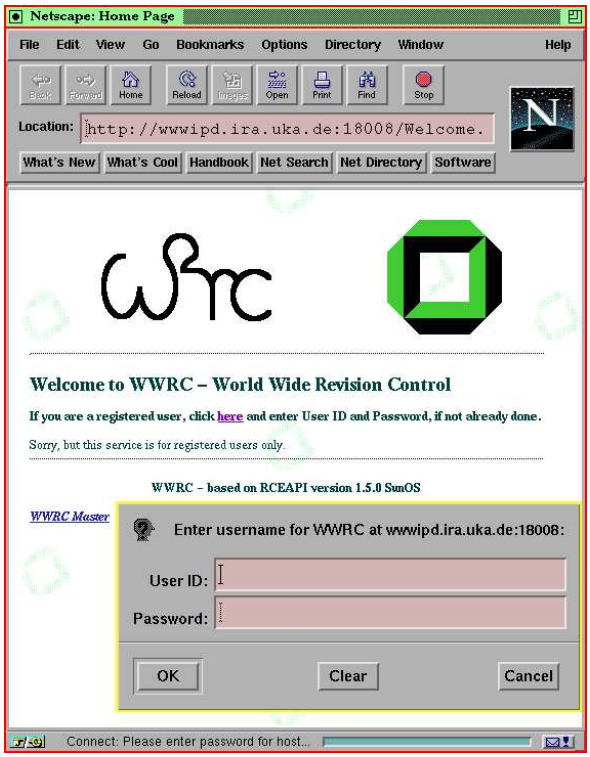

Abbildung 5: Einloggen in WWRC

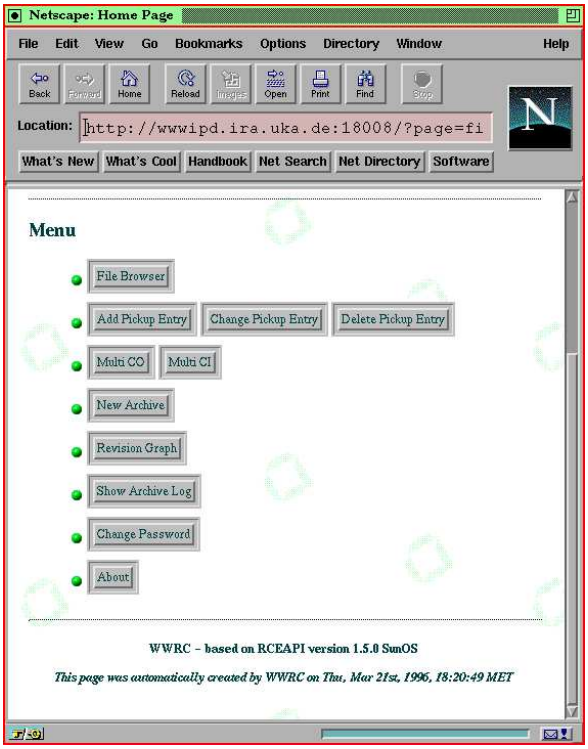

Abbildung 6: Hauptmenü

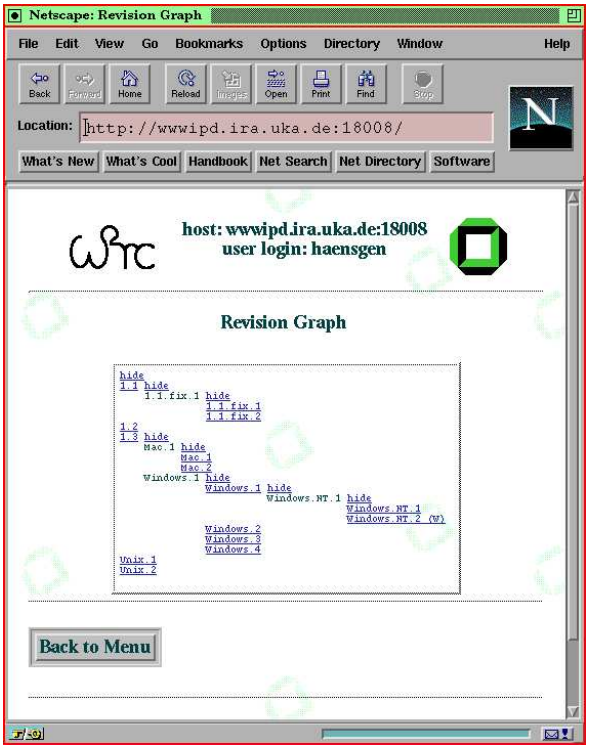

Abbildung 7: Revisionsgraph eines Archivs

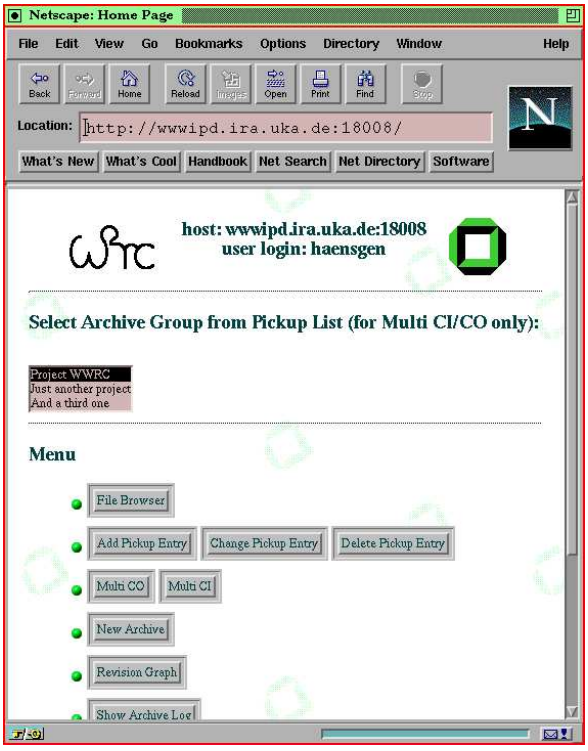

Abbildung 8: Auswählen einer Dateigruppe

Abbildung 9: Erfolgreiches Multi-CO

bemüht sich vielmehr, unabhängig von der Wahl eines Browsers volle Funktionalität zu bieten. Zwar läuft auch WWRC nicht mit jedem Browser; die aktuell verfügbaren Versionen von Mosaic beispielsweise haben Probleme mit der Darstellung von Tabellen. WWRC verwendet aber im Gegensatz zur Integrity Engine keine Mechanismen, die nicht Teil allgemein akzeptierter Protokolle sind[8].
Der in der Integrity Engine verfolgte Ansatz erfordert einen beträchtlichen Einsatz von Software auf der Client-Seite, was die Portierung auf andere Plattformen erschwert. WWRC hingegen benötigt dort nur die zwei kleinen, portablen Perl-Skripts. Laut MKS [17] unterstützt die Integrity Engine ebenso wie WWRC Versionskontrolle; inwieweit Konfigurationsmanagement unterstützt wird, geht hieraus nicht hervor.

5.2 BSCW

BSCW (vgl. [19], [20]) ist ein auf CSCW [18] aufbauendes System, das über das WWW einer Gruppe von Benutzern einen gemeinsamen Arbeitsbereich (*shared workspace*) zur Verfügung stellt.
Die Philosophie von BSCW ist es, Dateien (bzw. *Objekte*) ähnlich einem FTP-Server in hierarchisch gegliederten Verzeichnissen zu präsentieren. Diesen Dateien werden die jeweils auf sie anwendbaren Operationen an die Seite gestellt, zumeist in Form von Schaltflächen, die man anklicken kann; Formulare werden, soweit möglich, vermieden. Die Verwendung von Schaltflächen anstelle von Formularen erschwert jedoch die Parametrisierung von Operationen, etwa die Auswahl einer Revision aus einem Archiv, zumindest solange die Anzahl der Schaltflächen überschaubar bleiben soll.
BSCW basiert auf der Basis eines gemeinsamen Arbeitsbereiches; Sandboxen[9] werden offenbar nicht unterstützt. Wenn daher ein Benutzer eine Änderung im Arbeitsbereich vornimmt, sind alle Benutzer davon unmittelbar betroffen; dieses Konzept erschwert eine geregelte Revisions-, geschweige denn Konfigurationskontrolle. WWRC hingegen erweitert das revisionsorientierte RCE um das Konzept der Pickup-Liste (vgl. 2.4); dadurch erhält jeder Benutzer einen eigenen, individuellen Arbeitsbereich. Eine benutzerindividuelle Zusammenstellung von Objekten ist bei BSCW im Gegensatz zu WWRC nicht von vornherein vorgesehen. Ein automatisiertes Ein- oder Auschecken einer Menge von benutzerspezifischen Objekten wird daher nicht unmittelbar

[8] Daß das Protokoll für HTML/3.0 [10] derzeit den Status „expired“ genießt, ändert nichts daran, daß die dort eingeführten Tabellen ein allgemein akzeptierter Standard sind.

[9] Eine *Sandbox* stellt i.Ggs. zu einem gemeinsamen Arbeitsbereich einen individuellen Arbeitsbereich zur Verfügung.

unterstützt. Im Konzept von BSCW ist ferner keine Möglichkeit zu erkennen, ein Verzeichnis auf einen früheren Stand zurückzusetzen, um eine alte Konfiguration zu restaurieren. Dies ist beim aktuellen Stand von WWRC auch noch nicht implementiert, aber als zukünftige Funktion vorgesehen. Eine einfache Versionshaltung für einzelne Objekte kann mit dem Konzept von BSCW unterstützt werden, indem einzelnen Objekten Operationen in Form von Schaltflächen zur Seite gestellt werden, mit denen ein Check-in oder Check-out ausgeführt werden kann.

6 Weiterführende Arbeiten und Ausblick

Durch den nicht unbegrenzt dehnbaren Zeitrahmen für die Entwicklung von WWRC mußten einige Aspekte unberücksichtigt bleiben. Die nachfolgend aufgeführten offenen Punkte stellen jedoch keine grundsätzlichen Probleme dar; die zu ihrer Realisierung notwendigen Konzepte sind hinreichend klar. Vielmehr ist es eine Frage der Zeit und des Aufwandes, welche der folgenden Aspekte in eine zukünftige Implementierung mit einfließen sollen.
Im einzelnen lassen sich ohne Anspruch auf Vollständigkeit aufführen:

- WWRC ist, wie in Abschnitt 3.1 erläutert, in der vorliegenden Revision noch nicht als Dämon realisiert. Auf die hierfür notwendigen Änderungen wurde in diesem Abschnitt ebenfalls kurz eingegangen.

- Die Übertragung von Daten zwischen Client und WWRC-Server erfolgt bislang unverschlüsselt. Zwar ist eine Authentifikation bzw. Autorisierung mittels Paßwort gemäß dem in HTTP/1.0 definierten Basic Protection Scheme implementiert (vgl. 2.3); die Übertragung der Daten einschließlich des Paßwortes ist jedoch nicht gegen Lauschangriffe (*snooping*) geschützt. Aufgrund der vollständigen Kontrolle über die Transferdaten sowohl auf Client- wie auch auf Server-Seite stehen hier aber alle erdenklichen Schutzmaßnahmen offen. Zudem gibt es standardisierte Protokolle zur Autorisierung bzw. Authentifikation, die auf dem Basic Protection Scheme aufbauen und zusätzliche Sicherheiten bieten, so z.B. Verfahren mit öffentlichem Schlüssel (*public key*).

- Mit TCP/IP basiert WWRC auf einem Transportprotokoll, das bereits über rudimentäre Transportsicherungsmaßnahmen verfügt. Dennoch kann es insbesondere bei umfangreichen Datenübertragungen durchaus sinnvoll sein, zusätzliche Aufsetzpunkte zu definieren, bei denen im Falle einer unterbrochenen Verbindung die Übertragung wiederaufgenommen werden kann. Eine derartige Maßnahme läßt eine erhöhte Datensicherheit bei störanfälligem oder überlastetem Netzbetrieb erhoffen.

- Wenngleich *Java* [21] erst vor kurzer Zeit Einzug in das WWW gehalten hat, wird es von zunehmend mehr Browsern unterstützt und gilt schon jetzt als de-facto-Standard. Java bietet die Möglichkeit, plattformunabhängigen Programmcode vom Server zum Client zu schicken und dort ausführen zu lassen. Für WWRC ist hierfür insbesondere der – wenn auch aus Sicherheitsgründen nur eingeschränkt mögliche – Zugriff auf Dateien beim Client von Interesse. Es bleibt zu prüfen, ob auf die Hilfsapplikationen von WWRC gänzlich verzichtet werden kann. In diesem Falle wäre eine vollständige Client-seitige Plattformunabhängigkeit erreicht, solange die Verfügbarkeit eines Java-tauglichen Browsers gesichert ist.

Einige Aspekte bedürfen jedoch noch einer näheren wissenschaftlichen Untersuchung. Hierzu gehören insbesondere die folgenden beiden Punkte, die als konzeptuelle Herausforderung für zukünftige Versionen von WWRC bestehen bleiben:

- Wie in 2.1 ausgeführt, unterstützt WWRC nur Revisionskontrolle, jedoch kein Konfigurationsmanagement. Hier kann das Wissen und die Erfahrung, die im Umgang mit anderen Systemen zur Revisions- bzw. Konfigurationskontrolle gesammelt wurden, in die zukünftige Entwicklung mit einfließen.
- Eine Verfeinerung der Zeitstempel-Überwachung, wie in 2.3 angedeutet, kann zum Ziele einer flexibleren Bedienbarkeit von WWRC durchgeführt werden. Es wird dadurch dem Benutzer möglich, auch von alten WWW-Seiten aus den Dialog wiederaufzunehmen, solange dies nicht zu Inkonsistenzen führt. Insbesondere das Erkennen solcher Inkonsistenzen und die Einleitung geeigneter Gegenmaßnahmen bedarf dabei noch einer genaueren Untersuchung.

7 Zusammenfassung

Mit der Entwicklung und Implementation des Prototyps von WWRC konnte gezeigt werden, daß trotz einiger technischer Probleme, wie sie etwa durch Beschränkungen in den Protokollen HTTP und HTML herrühren, die Realisierung eines verteilten Systems zur Revisionskontrolle unter Einbindung in das World Wide Web möglich ist. Eine vollständigere und detailliertere Ausarbeitung der Implementation, die insbesondere auch Mechanismen zur Konfigurationskontrolle umfaßt, bleibt als zukünftige Herausforderung bestehen.

8 Anhang

Die folgende Tabellen listen die Module und Interfaces von WWRC auf, um einen Eindruck der Struktur und des Umfangs des Projekts zu vermitteln. Sie enthalten auch die Größe jedes Moduls bzw. Interface sowie eine Aufzählung importierter Module.

Interface/Module	Größe (Bytes)	IMPORT	Funktion
Basis-Module			
`Cfg.i3`	2260	(keine);	Einlesen der Konfigurations-Datei; Bereitstellen globaler Konfig.-Variablen
`Cfg.m3`	7154	`Log, Document, Scanner, EnvList, RCEAPI;`	Einlesen der Konfig.-Datei
`Log.i3`	172	(keine);	Log-Datei für Überwachungs- und Debugging-Zwecke
`Log.m3`	3001	`Cfg, Msg, Exit;`	
`Msg.i3`	365	(keine);	Formatierte Ausgabe von Warnhinweisen und Fehlermeldungen
`Msg.m3`	2187	`Lng, Exit;`	
`Exit.i3`	66	(keine);	Kontrollierter (Not-)Abbruch des Prog.; Freigabe evtl. noch gesperrter Archive
`Exit.m3`	811	`Log, UserEnv, RCEAPI, RCEUtils;`	
`Lng.i3`	326	(keine);	Sprachmodul (nur ansatzweise implementiert)
`SysCall.i3`	711	(keine);	Modul für Aufruf von System-Befehlen
`SysCall.m3`	5124	`Log, Document;`	

Interface/Module	Größe (Bytes)	IMPORT	Funktion
Datenstrukturen			
AVL.i3	1512	(keine);	allg. AVL-Baum
AVL.m3	12365	Document;	
IdAVL.i3	389	AVL;	AVL-Baum über Zeichenketten
IdAVL.m3	1482	Document, AVL;	
EnvList.i3	889	(keine);	Verwaltung von Umgebungsvariablen
EnvList.m3	4831	Document, AVL, IdAVL;	
SysEnv.i3	50	(keine);	allgemeine Umgebungsvariablen
SysEnv.m3	367	Cfg, Log;	
UserEnv.i3	802	Scanner;	benutzerspezif. Umgebungsvariablen; werden von/auf Benutzer-Datei gelesen/geschrieben
UserEnv.m3	23568	Cfg, Log, RCEAPI, Document, PickupList, Scanner, EnvList;	
Scanner.i3	1750	(keine);	Zerteiler für allgemeine Zwecke
Scanner.m3	25064	Log, Document;	
RCE			
rce.i3	9509	(keine);	Interface für externen Import von rce
TypeCastCM3.i3	1253	(keine);	Typ-Konvertierungen C $\leftrightarrow$ Modula-3
TypeCastCM3.m3	2507	(keine);	
RCEAPI.i3	14301	(keine);	wie rce.i3, jedoch mit Modula-3-Typen und Exceptions
RCEAPI.m3	33100	TypeCastC3, rce;	
RCEUtils.i3	1582	RCEAPI, PickupList;	Erweiterte RCE-Funktionalität (z.B. MultiRCERevOpen/Close)
RCEUtils.m3	10667	Cfg, Log, RCEAPI, Document, PickupList;	
RevGraph.i3	1408	RCEAPI, RCEUtils, PickupList;	Aufbau eines RCE-Graphen von Benutzerdatei aus bzw. über RCEAPI
RevGraph.m3	21188	Log, Cfg, RCEAPI, UserEnv, Document, RCEUtils, PickupList;	
PickupList.i3	2434	RCEAPI;	Verwaltung der Pickup-Liste
PickupList.m3	27630	Log, Cfg, RCEAPI, Document, Scanner;	

Interface/Module	Größe (Bytes)	IMPORT	Funktion
HTTP			
`ReqFields.i3`	890	(keine);	Verwaltung der Daten-Felder einer HTTP-Anfrage
`ReqFields.m3`	3098	`EnvList, Document;`	
`Response.i3`	1222	(keine);	Verwaltung der Daten-Felder einer HTTP-Antwort
`Response.m3`	5554	`Cfg, Log, Document, EnvList;`	
`Authorization.i3`	124	(keine);	Zugriffskontrolle
`Authorization.m3`	9246	`Log, UserEnv, Response, Document;`	
`HTTP.i3`	826	(keine);	Syntaktische Analyse einer HTTP-Anfrage
`HTTP.m3`	12480	`Log, Document, ReqFields, Response, Exit;`	
HTML			
`FileTransfer.i3`	465	(keine);	Fileserver-Funktion von WWRC
`FileTransfer.m3`	8429	`Log, Exit, Document, Response;`	
`HTML.i3`	3758	(keine);	Erzeugung von HTML-Tags
`HTML.m3`	32486	`Cfg, Log, Document;`	
`HTMLAccessories.i3`	795	(keine);	Erzeugung WWRC-spezifischer HTML-Bausteine
`HTMLAccessories.m3`	9306	`HTML, Cfg, Log, RCEAPI, Document, ReqFields, UserEnv;`	
`FileManager.i3`	145	(keine);	Bausteine für FileBrowser (nur ansatzweise implementiert)
`FileManager.m3`	616	`HTML;`	
`Document.i3`	5911	`RCEAPI, PickupList;`	Erzeugung WWRC-spezifischer HTML-Seiten
`Document.m3`	62943	`Log, Cfg, Exit, RCEAPI, RCEUtils, FileTransfer, UserEnv, RevGraph, PickupList, Response, ReqFields;`	

Interface/Module	Größe (Bytes)	IMPORT	Funktion
Main			
`PP.i3`	77	(keine);	Syntakt./semant. Analyse der Kommandozeilen-Parameter
`PP.m3`	926	`Cfg, Exit;`	
`wwrc.m3`	52489	`Cfg, SysEnv, PP, Log, RCEAPI, UserEnv, FileTransfer, RevGraph, Document, HTTP, RCEUtils, PickupList, ReqFields, Authorization, Exit, SysCall;`	globale Kontrollflußsteuerung
Summe	452297		

Literatur

[1] J. Reuter, S.U. Hänssgen, J.J. Hunt, W.F. Tichy: *Distributed Revision Control via the World Wide Web.*
`http://wwwipd.ira.uka.de/~reuterj/wwrc.ps.gz`

[2] J.J. Hunt, K.-P. Vo, W.F. Tichy: *An Empirical Study of Delta Algorithms.*
`http://wwwipd.ira.uka.de/~jjh/delta.ps.gz`

[3] Samuel P. Harbison: *Modula 3.* Prentice Hall, 1992

[4] Walter F. Tichy: *RCS: A Revision Control System.* Integrated Interactive Computing Systems. North-Holland Publishing Co., 1983

[5] Xcc Software Technology Transfer GMBH: *RCE - Introduction and Reference Manual*, API Revision 1.4.5 edition, 1995.

[6] A. Luotonen, H. Frystyk, T. Berners-Lee: *W3C HTTPd.*
`http://www.w3.org/pub/WWW/Daemon/`

[7] *NCSA HTTPd.* NCSA HTTPd Development Team, Software Development Group of the National Center for Supercomputing Applications at the University of Illinois at Urbana – Champaign, Illinois, USA.
`http://hoohoo.ncsa.uiuc.edu/docs/Overview.html`

[8] T. Berners-Lee, R. Fielding, H. Frystyk: *Hypertext Transfer Protocol – HTTP/1.0.* HTTP Working Group, Internet-Draft, February 19, 1996.
`http://www.w3.org/pub/WWW/Protocols/HTTP/1.0/spec.ps.gz`

[9] T. Berners-Lee, D. Conolly: *Hypertext Markup Language – 2.0.* HTML Working Group, Internet-Draft, September 22, 1995.
`http://www.w3.org/pub/WWW/MarkUp/html-spec/html-spec.ps.gz`

[10] Dave Raggett: *HyperText Markup Language Specification Version 3.0.* HTML Working Group, Internet-Draft, 28th March 1995.
`http://www.w3.org/pub/WWW/MarkUp/html3/html3.txt.Z`

[11] *The Common Gateway Interface – CGI/1.1.* NCSA HTTPd Development Team, Software Development Group of the National Center for Supercomputing Applications at the University of Illinois at Urbana – Champaign, Illinois, USA.
`http://hoohoo.ncsa.uiuc.edu/cgi/overview.html`

[12] David H. Crocker: *RFC 822: Standard for the Format of Arpa Internet Text Messages.*
`http://www.w3.org/hypertext/WWW/Protocols/rfc822/Overview.html`

[13] N. Freed: *MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies.*
`http://www.cis.ohio-state.edu/htbin/rfc/rfc1521.html`

[14] T. Berners-Lee: *Uniform Resource Locators (URL).*
`http://www.w3.org/hypertext/WWW/Addressing/rfc1738.txt`

[15] R. Fielding: *Relative Uniform Resource Locators.*
`http://www.w3.org/hypertext/WWW/Addressing/rfc1808.txt`

[16] SCCS
http://www-h.eng.cam.ac.uk/help/tpl/languages/sccs.html

[17] Mortice Kern Systems (MKS) Inc.: *Integrity Engine Data Sheet.*
http://www.mks.com/netscape/webscm/ie_spec.htm

[18] Volker Paulsen: *CSCW Research Group Homepage.*
http://orgwis.gmd.de/

[19] R. Bentley, T. Horstmann, K. Sikkel, J. Trevor: *Supporting collaborative information sharing with the World-Wide Web: The BSCW shared workspace system.* Proceedings of the 4th International WWW Conference, Boston, MS, 1995

[20] R. Bentley, W. Appelt: *CSCW Research Group: BSCW Project.*
http://orgwis.gmd.de/BSCW/

[21] James Gosling, Henry McGilton: *The Java Language Environment*, 1995.
http://www.javasoft.com/documentation.html