Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

API	……	Application Programming Interface
CASE	……	Computer Aided Software Engineering
CFML	……	Cold Fusion Markup Language
CGI	……	Common Gateway Interface
CORBA	……	Common Object Request Broker Architecture
DMZ	……	Demilitarized Zone
DNS	……	Domain Name Service
ERM	……	Entity Relationship Modeling
HABIT	……	Hagener Betrieb für Informationstechnologie
HTML	……	Hypertext Markup Language
HTTP	……	Hypertext Transfer Protokoll
IIS	……	Internet Information Server
IT	……	Informationstechnologie
MIME	……	Multipurpose Internet Mail Extension
OSCI	……	Online Services Computer Interface
Perl	……	practical extraction and report language
QM	……	Qualitätsmanagement
QS	……	Qualitätssicherung
RDBMS	……	Relationales Datenbank Management System
SGML	……	Standard Generalized Markup Language
SMTP	……	Simple Mail Transfer Protocol
SQL	……	Structured Query Language
TCP/IP	……	Transmission Control Protocol/Internet Protocol
UML	……	Unified Modeling Language
URL	……	Uniform Resource Locator
WWW	……	World Wide Web
XML	……	eXtensible Markup Language

1 Einleitung

1.1 Motivation

Das generelle Interesse an der Automatisierung von technischen Systemen und obendrein die Möglichkeit der Wissensaneignung für die berufliche Tätigkeit waren für den Autor der Anlass, sich im Rahmen einer Diplomarbeit mit dem gewählten Thema auseinander zu setzen.

Die Entwicklung einer datenbankgestützten Anwendungssoftware zur automatisierten Überprüfung eines Internetportals, unter Berücksichtigung der entsprechenden Randgebiete, bot die Möglichkeit sich mit einem breiten Spektrum von praxisrelevanten und aktuellen Themen zu beschäftigen.

Zu den betrachteten Hauptgebieten zählten vor allem die Bereiche:

- Grundlagen des Internet
- Softwareentwicklung mit der Programmiersprache Perl/TK und
- Datenbankentwicklung mit dem relationalen Datenbanksystem MySQL

Darüber hinaus bot das Thema die Möglichkeit, einen Einstieg in das Thema Software-Qualitätssicherung zu finden.

1.2 Zielsetzung der Arbeit

Ziel der Diplomarbeit ist die Entwicklung einer datenbankgestützten Anwendungssoftware, deren Aufgabe es ist, das Internetportal der Stadt Hagen hinsichtlich bestimmter Grundfunktionen automatisiert zu überprüfen. Aufgrund der funktionalen Ausrichtung handelt es sich bei dem zu entwickelnden Prototypen um eine Software, die im Bereich des Software-Qualitätsmanagement angesiedelt ist und den so genannten *Blackbox-Verfahren*[1] zugeordnet werden kann.

Basierend auf dem *Pflichtenheft*[2], welches in Zusammenarbeit mit dem Hagener Betrieb für Informationstechnologie (kurz HABIT) erstellt wurde, sollen in dieser Diplomarbeit die Entwicklungsphasen Entwurf und Implementierung anhand von ausgewählten *Use Cases*[3] exemplarisch bearbeitet werden.

Der entwickelte Prototyp soll mit den entsprechenden Komponenten beim HABIT zum Einsatz kommen und dadurch einen Beitrag zur Qualitätssicherung des Internetangebots der Stadt Hagen leisten.

[1] Das Blackbox-Verfahren ist ein Begriff aus dem Software-Qualitätsmanagement und wird im Kapitel 3 „Vermittlung von relevanten Grundlagen“ näher erläutert.
[2] Die genauen Anforderungen an die Anwendungssoftware können dem Pflichtenheft (Anhang A) entnommen werden.
[3] Der Begriff „Use Case“ stammt aus dem Bereich der Unified Modeling Language (UML) und stellt eine Möglichkeit dar, die Funktionen eines Softwaresystems zu beschreiben. Dazu bedient man sich in der Regel eines so genannten „Use Case-Diagramms“.

1.3 Hagener Betrieb für Informationstechnologie (HABIT)

1.3.1 Geschäftsbereich Technischer Betrieb

Der HABIT ist Dienstleister für die Stadt Hagen und die Kooperationspartner im Ennepe-Ruhr-Kreis. Die Angebotspalette des HABIT umfasst die Bereitstellung der IT- und Kommunikationsstrukturen und die Betreuung von circa 150 kommunalen *Fachverfahren*[4] sowie die entsprechende IT- und Organisationsberatung.

Eines der wichtigsten Projekte des HABIT ist die Betreuung des eGovernment-Portals der Stadt Hagen - mit dem Namen „Virtuelles Rathaus".

Unter Electronic Government (eGovernment) versteht Lucke *„die Abwicklung geschäftlicher Prozesse im Zusammenhang mit Regieren und Verwalten mit Hilfe von Informations- und Kommunikationstechniken über elektronische Medien"*[5].

Für dieses Projekt übernimmt der HABIT die Projektleitung, den Betrieb und die Weiterentwicklung.

Um den technischen Betrieb des Virtuellen Rathauses und somit die Verfügbarkeit der im Internet angebotenen Dienste und Produkte permanent zu gewährleisten, werden bestimmte Grundfunktionalitäten periodisch (täglich, wöchentlich und monatlich) getestet. Diese Tests werden von den Mitarbeitern des HABIT mit Hilfe von abzuarbeitenden Listen manuell durchgeführt.

[4] Fachverfahren ist die Bezeichnung für eine Anwendungssoftware für verwaltungsspezifische Aufgaben im öffentlichen Dienst.
[5] Siehe (**Lucke 2000**, S. 1)

1.3.2 Motivation für den HABIT

Die manuelle Vorgehensweise bei der Testdurchführung zur Gewährleistung der einwandfreien Funktionalität des Virtuellen Rathauses ist mit folgenden Problemen verbunden:

- Hoher Zeit- und Kostenaufwand (Verschwendung von Personalressourcen)
- Demotivation der Mitarbeiter durch monotone Tests
- Keine Tests in arbeitsfreien Zeiten
- Geringe Testgeschwindigkeit
- Die Testdurchführung ist fehleranfällig und nicht vollständig reproduzierbar
- Die Testfrequenz, der Umfang und die Komplexität kann bei manuellen Tests nicht „beliebig“ erhöht werden

Diese Probleme veranlassten den HABIT sich für die Entwicklung eines Prototyps zu entscheiden, der diese Probleme ganz beziehungsweise teilweise lösen kann.

1.4 Vorgehensweise und Gliederung

Zur Erreichung der geplanten Ziele wurde eine Gliederung der Diplomarbeit in folgende Kapitel vorgenommen:

Im Anschluss an diesen Einführungsteil werden im zweiten Kapitel die wichtigsten Ergebnisse des Pflichtenheftes zusammengefasst, um die Ausrichtung der Anwendungssoftware und die damit verbundenen offenen Problembereiche zu erkennen.

Im dritten Kapitel werden die notwendigen Grundlagen vermittelt, um die Problemstellungen, die bei den weiteren Entwicklungsphasen zu berücksichtigen und zu bearbeiten sind, besser verstehen zu können.

In diesem Kapitel wird auf den Aspekt der Software-Qualitätssicherung und die Grundlagen des Internets, welche die Basis für das Virtuelle Rathaus bilden, eingegangen. Weiterhin wird die Systemarchitektur des Virtuellen Rathauses und die Funktionsweise des ColdFusion Servers, der das Kernsystem des Virtuellen Rathauses bildet, erklärt.

Eine Aufstellung der Vor- und Nachteile der Programmiersprache Perl und eine Vorstellung des relationalen Datenbankmanagement-Systems MySQL™ bilden den Abschluss dieses Kapitels.

Im vierten Kapitel erfolgt die Durchführung der Entwurfsphase. Unter Verwendung verschiedener Entwurfsmethoden werden die Daten, die Funktionen und die Benutzungsoberfläche der Anwendungssoftware modelliert.

Die Vorstellung der Systemarchitektur der Anwendungssoftware bildet das fünfte Kapitel. Hier wird auf die Architektur und die integrierten Funktionen, beziehungsweise Module eingegangen.

Im sechsten Kapitel wird exemplarisch die Implementierung von wichtigen Funktionen und Konzepten erläutert. Darüber hinaus werden die verwendeten Implementierungswerkzeuge angesprochen.

Das siebte Kapitel bildet den Schlussteil der Diplomarbeit, hier werden die Erkenntnisse zusammengefasst und ein Fazit gezogen.

Im Anhang befinden sich das Pflichtenheft und die SQL-Anweisungen, die zur Erzeugung der Datenbank notwendig sind. Die weiteren Quelltexte und die frei verfügbaren Softwareprodukte sind, neben einer Installationsanleitung, auf der mitgelieferten CD zu finden.

2 Zusammenfassung der Definitionsphase

Die Hauptfunktionen der Anwendungssoftware *VirTest*[6], welche genauso wie die weiteren Rahmenbedingungen im Pflichtenheft explizit definiert sind, lassen sich anhand des folgenden Basis-Use-Case-Diagramms (Abbildung 1) übersichtlich darstellen.

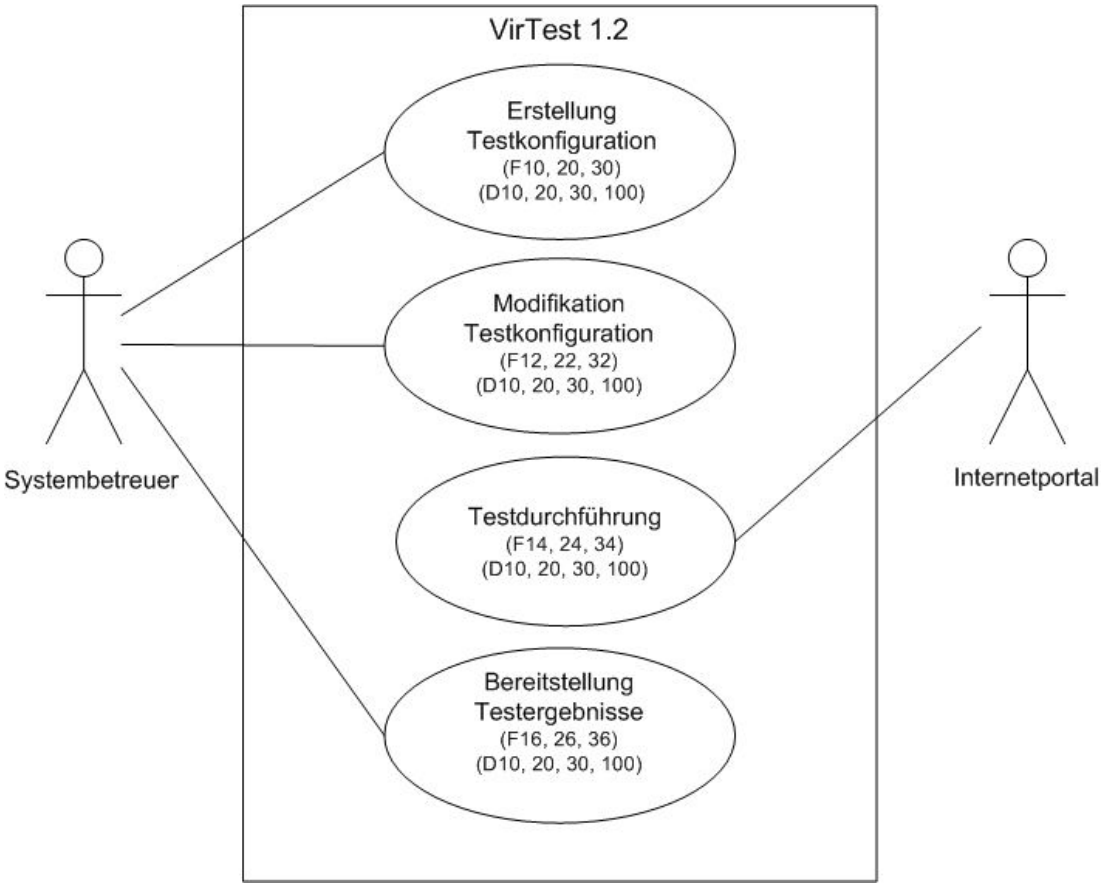


Abbildung 1: Basis-Use-Case Diagramm VirTest Version 1.2

Es ist offensichtlich, dass ein großer Teil der Funktionalität der Anwendungssoftware VirTest im Bereich der Administration der Testdaten, also der Erstellung und Modifikation der Daten liegt. Der zweite Hauptteil liegt in der Testdurchführung und der dritte Teil in der Bereitstellung der Testergebnisse.

[6] Um dem in den nachfolgenden Ausführungen nicht mehr mit den Begriffen „Softwareprodukt“ oder Anwendungssoftware arbeiten zu müssen, wird an dieser Stelle der Name „VirTest“ mit der Versionskennzeichnung 1.2 eingeführt. Der Name setzt sich aus den Begriffen Virtuelles Rathaus und Test zusammen. Somit kennzeichnet diese Bezeichnung den Kontext und die Zielsetzung des Produkts. Die Versionshistorie kann ebenfalls dem Pflichtenheft im Anhang A entnommen werden.

Im Rahmen der Planungsphase wurden in Zusammenarbeit mit dem HABIT mit Hilfe einer *Gewichtungsmethode*[7] - unter Berücksichtigung der Parameter Dauer, Wichtigkeit und Realisierbarkeit - die einzelnen manuellen Funktionstests bewertet.

Folgende Tests des Virtuellen Rathauses wurden als besonders wichtig identifiziert und können daher dem Basis Use Case „Testdurchführung“ zugeordnet werden.

- Kontrolle von Weboberflächen
- Kontrolle von HTML-Links
- Kontrolle des einwandfreien Transaktionsvermögens

Diese Tests bzw. Kontrollen des Virtuellen Rathauses sollen mit Hilfe von VirTest automatisiert durchgeführt werden. Dazu müssen zunächst die Testparameter konfiguriert und persistent gespeichert werden. Daher wird eine Speicherung in einer Datenbank angestrebt.

Eine Unterstützung der Benutzer bei der Administration der einzelnen Testparameter und der Testdurchführung wird weiterhin mit Hilfe eines Graphical User Interfaces erreicht. Hier können die Testparameter in den dafür vorgesehenen Eingabefeldern administriert und weitere Funktionen hinsichtlich der Testdurchführung ausgeführt werden.

Um zunächst einen Überblick über den Kontext von VirTest und die damit verbundenen Schnittstellen (Interfaces) sowohl zur Umwelt als auch innerhalb der Software zu identifizieren, dient die nachfolgende Abbildung 2.

[7] Eine Beschreibung dieser Gewichtungsmethode kann der Projektarbeit entnommen werden, welche dieser Diplomarbeit zugrunde liegt.

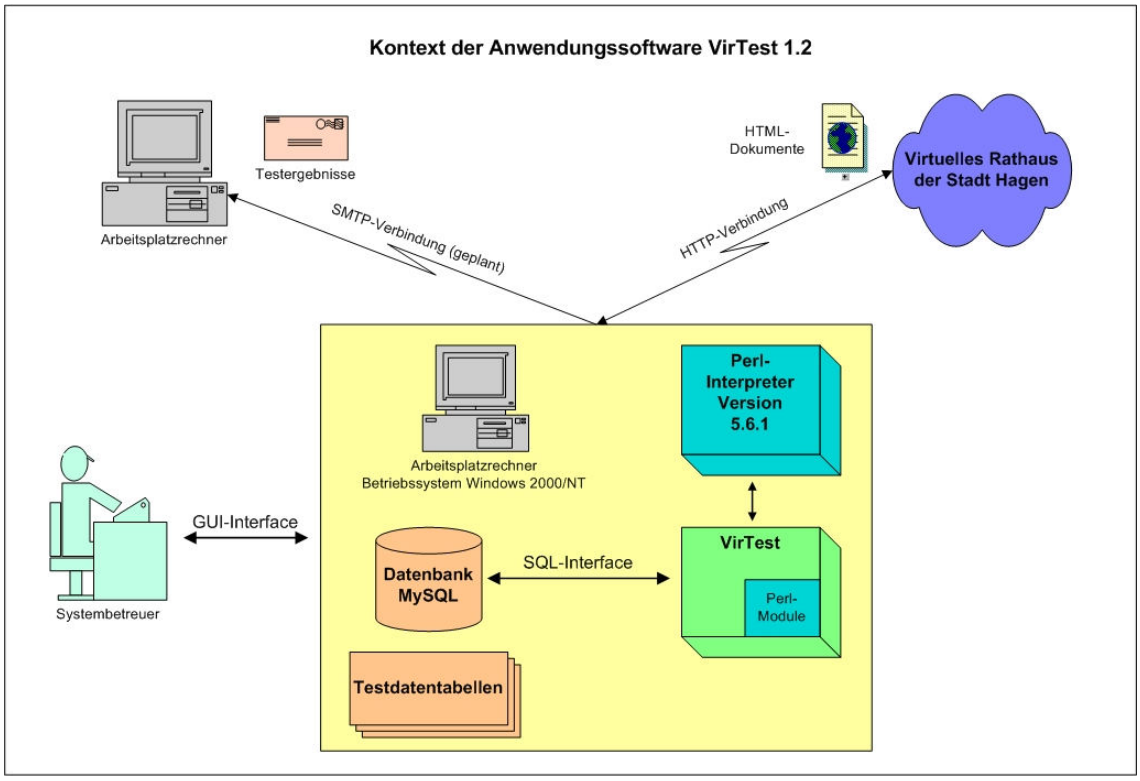


Abbildung 2: Kontext Anwendungssoftware

Anhand der dargestellten Abbildung können folgende Interfaces zur Kommunikation erkannt werden, auf welche im Kapitel 5 „Systemarchitektur VirTest“ näher eingegangen wird:

a) Interfaces zur Umwelt:

- GUI-Interface (Mensch-Maschine)
- Internet-Verbindung
- SMTP-Verbindung (geplant)

b) Interne-Interfaces:

- Datenbankinterface

3 Vermittlung von relevanten Grundlagen

3.1 Eingrenzung der Grundlagenbereiche

Eine Betrachtung der Funktionstests zeigt, dass ein Teil der Themen in engem Zusammenhang mit den Eigenschaften, Konzepten und Funktionen des Internets steht. Weitere wichtige Gebiete, die zum Verständnis und zur Bearbeitung der Diplomarbeit dienen, sind die verwendete Programmiersprache und die genutzte Datenbank.

Daher wird versucht in den folgenden Unterkapiteln einen Einblick in diese Bereiche zu verschaffen. Zuvor wird VirTest noch unter dem Aspekt des *Software-Qualitätsmanagements*[8] betrachtet.

3.2 Software-Qualitätsmanagement

Aufgrund der funktionalen Ausrichtung von VirTest leistet die Software einen Beitrag zur Qualitätssteigerung und ist daher im Bereich des Software-Qualitätsmanagements angesiedelt. Es soll daher in den folgenden Unterkapiteln auf die Klassifizierung der Software eingegangen werden.

3.2.1 Begriffe und Klassifizierung

Das Testen des Internetportals der Stadt Hagen mit Hilfe von VirTest dient vorrangig zur Identifizierung von Defekten und deren Beseitigung durch die Mitarbeiter des HABIT, um so eine Steigerung der Softwarequalität zu erreichen.

Was bedeutet Softwarequalität?

Nach DIN ISO 9126 wird hierunter die *„Gesamtheit der Merkmale und Merkmalswerte eines Software-Produkts, die sich auf dessen Eignung beziehen, festgelegte oder vorausgesetzte Erfordernisse zu erfüllen*“[9] verstanden.

[8] In dieser Arbeit können nur die Aspekte des Software-Qualitätsmanagement angesprochen werden, die zur Klassifizierung der Anwendungssoftware VirTest notwendig sind. Eine detaillierte Bearbeitung des Themas ist in **[Balzert 1998**] zu finden.
[9] Siehe DIN ISO 9126

Dazu werden in der gleichen DIN 9126 folgende sechs Merkmale für Software-Produkte definiert, anhand derer eine Bewertung von Software möglich wird.

Dies sind die:

- Funktionalität
- Zuverlässigkeit
- Benutzbarkeit
- Effizienz
- Änderbarkeit
- Übertragbarkeit

Abhängig von der weiteren Unterteilung und Bewertung dieser Qualitätsmerkmale können diese bestimmten Qualitätsmodellen zugeordnet und entsprechende Qualitätsanforderungen aufgestellt werden.

Zur Sicherstellung der aufgestellten Qualitätsanforderungen kann sich dann bestimmter Qualitätssicherungsmaßnahmen bedient werden.

Grundsätzlich umfasst die Qualitätssicherung (QS) „*alle geplanten und systematischen Tätigkeiten, die innerhalb des Qualitätsmanagementsystems verwirklicht sind, und die wie erforderlich dargelegt werden, um angemessenes Vertrauen zu schaffen, dass eine Einheit die Qualitätsforderung erfüllen wird*“[10].

Dabei lassen sich die QS-Maßnahmen in konstruktive und analytische Maßnahmen unterscheiden.

Zu den konstruktiven Qualitätsmanagementmaßnahmen zählen zum Beispiel Methoden, Sprachen, Werkzeuge, Richtlinien, Standards und Checklisten, die dafür sorgen, dass das entstehende Produkt bzw. der Erstellungsprozess bestimmte Eigenschaften besitzt.

[10] Siehe DIN EN ISO 8402

Die nachfolgende Abbildung 3 differenziert die konstruktiven QS-Maßnahmen zusätzlich in technische und organisatorische Maßnahmen.

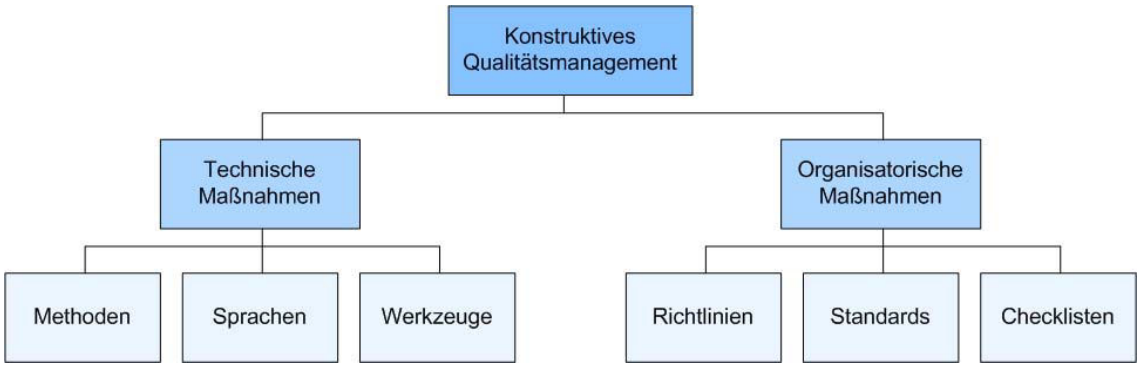


Abbildung 3: Maßnahmen zum konstruktiven Qualitätsmanagement[11]

Bei den analytischen Maßnahmen hingegen handelt es sich um diagnostische Mittel, die zur Ermittlung des erreichten Qualitätsniveaus des *Testobjekts*[12] und zur Identifizierung von Defekten dienen.

Das Ziel ist hier vorrangig die Prüfung und Bewertung des jeweiligen Testobjekts, in diesem Fall das Virtuelle Rathaus der Stadt Hagen. Daher kann VirTest dem analytischen Bereich zu geordnet werden.

Eine weitere Unterteilung des analytischen Qualitätsmanagements kann wiederum durch die Einteilung in analysierende und testende Verfahren vorgenommen werden. Die nachfolgende Abbildung 4 verdeutlicht diese Einteilung.

[11] Siehe (**Balzert 1998**, S. 280)
[12] Die Software-Komponente beziehungsweise das Programm oder System, welches getestet werden soll, wird als Testobjekt bezeichnet.

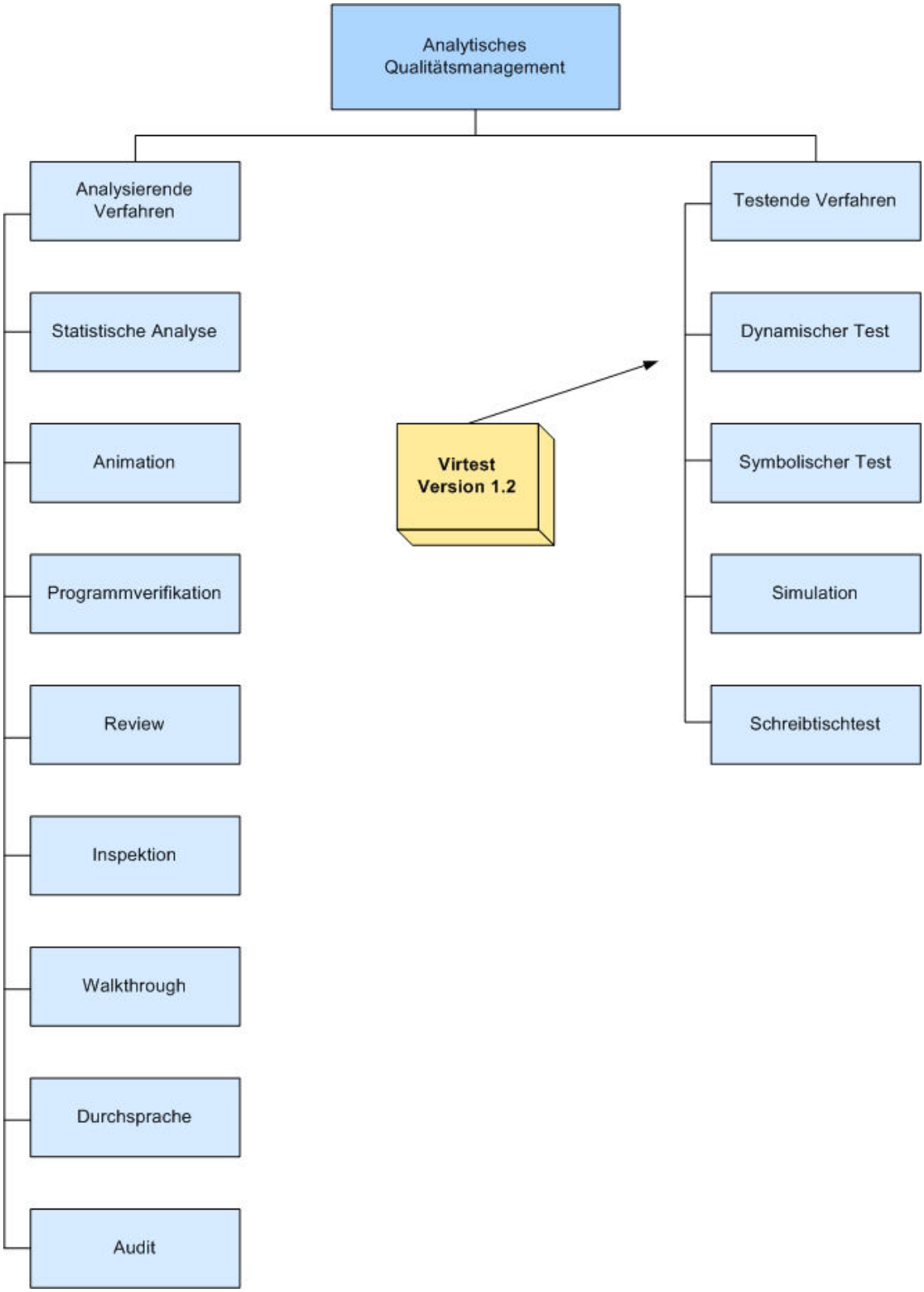


Abbildung 4: Maßnahmen zum analytischen Qualitätsmanagement[13]

[13] Vgl. (**Balzert 1998**, S. 281)

Nach Balzert lassen sich die analytischen Qualitätsmanagement-Verfahren folgendermaßen charakterisieren:

„Analysierende Verfahren sammeln gezielt Informationen über den Prüfling mit analytischen Mitteln, dass heißt unter Verzicht auf die dynamische Ausführung des Prüflings mit konkreten Eingaben.“[14]

Testende Verfahren führen stattdessen das Testobjekt mit Eingaben aus. Daher kann VirTest dem Bereich der dynamischen Tests zugeordnet werden.

Die gemeinsamen Merkmale der dynamischen Tests sind die folgenden:

- *Das ausführbare Testobjekt (das Virtuelle Rathaus) wird mit konkreten Eingabewerten versehen und ausgeführt.*
- *Das Testobjekt kann in der realen Umgebung getestet werden.*
- *Es handelt sich um ein Stichprobenverfahren, das heißt, die Korrektheit des getesteten Programms wird nicht bewiesen.*[15]

3.2.2 Black- und White Box-Verfahren

Die dynamischen Tests lassen sich erneut in eine Vielzahl von Testverfahren unterscheiden, deren gemeinsame Merkmale zuvor angesprochen wurden.

Zwei der wichtigsten Testverfahren dieser Klasse sind das Black-Box- und das White-Box-Testverfahren. Der Unterschied zwischen diesen beiden Verfahren liegt in der Ermittlung der *Testfälle*[16], die mit dem Testobjekt durchgeführt werden sollen.

[14] Siehe (**Balzert 1998**, S. 281)
[15] Vgl. (**Balzert 1998**, S. 396)
[16] Unter einem Testfall versteht man die Gesamtheit der Testdaten, die zur Ausführung des zu testenden Programms notwendig sind.

Die Abbildung 5 zeigt die Unterschiede dieser beiden Verfahren auf und klassifiziert VirTest abschließend.

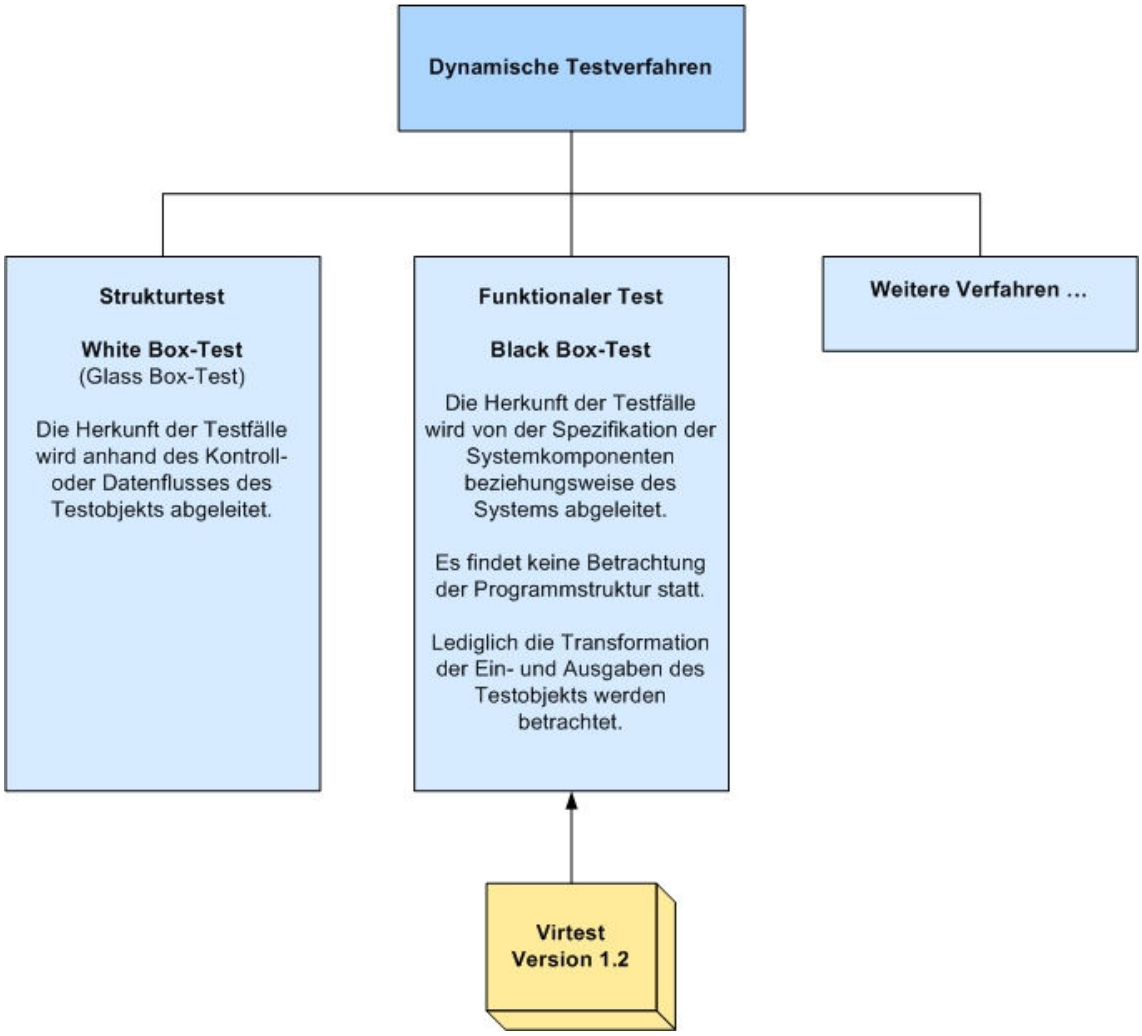


Abbildung 5: Differenzierung des Black- und White-Box-Testverfahrens

3.3 Internet – Das Netz der Netze

Die Anfänge des heutigen Internets beruhen auf eine Initiative des US-amerikanischen Verteidigungsministeriums in den 60er Jahren. Ziel war der Aufbau eines fehlertoleranten Rechnernetzes, das bei einem Ausfall von Rechnern und von Übertragungsleitungen, die Kommunikation zwischen den verbliebenen Rechnern weiterhin gewährleisten sollte.

Angefangen mit vier Rechnern, die ausschließlich mit militärischen Problemstellungen beschäftigt waren, wuchs das so genannte „Netz der Netze“ nach der Aufhebung der Beschränkung auf die Nutzung zu militärischen Aufgaben, auf heute allein in Deutschland *ca. 38 Millionen Internetnutzer*[17]an.

Diese Vielzahl von Rechnern wird mittels Einteilung in zugehörige Netze strukturiert (Abbildung 6).

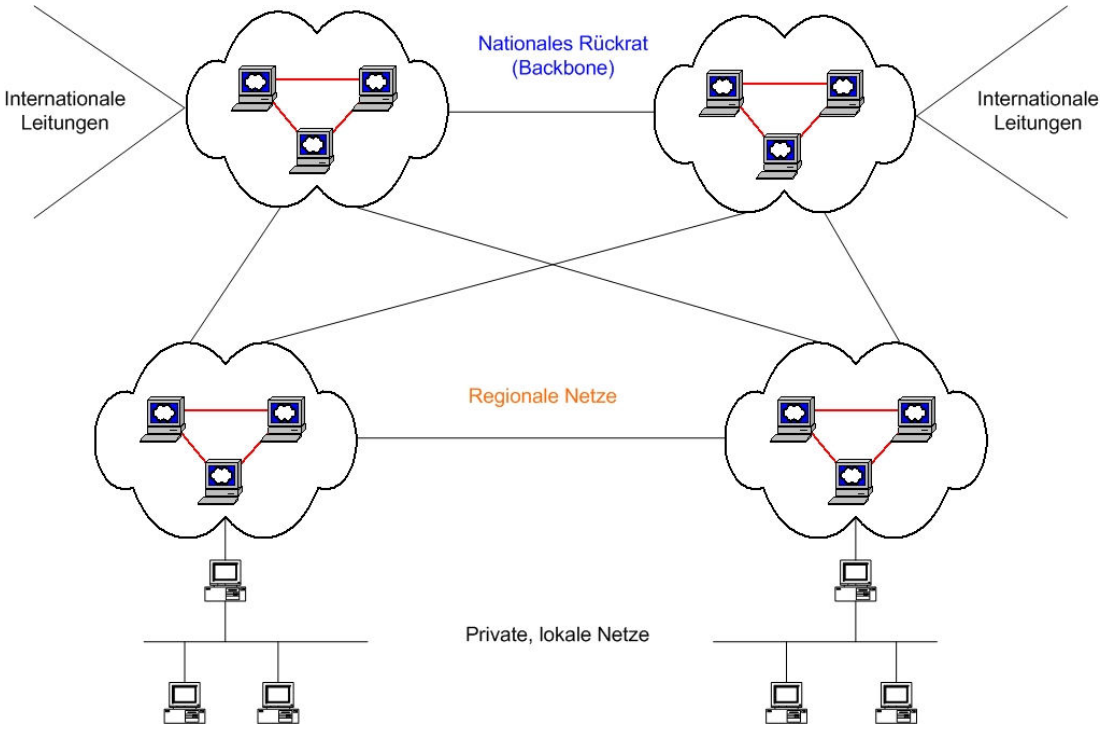


Abbildung 6: Struktur des Internets

[17] Vgl. (**Kahle 2004**, S. 5)

Die Gesamtheit dieser Netze, die über öffentliche Datennetze mit Wähl- oder Festverbindungen, Standleitungen, Satelliten, Funknetz und das Telefonnetz realisiert sind, bildet das eigentliche Internet.

Bei den Netzen handelt es sich vorrangig um lokale Netze von Hochschulen, Forschungseinrichtungen, Firmen und Behörden.

Diese zunächst wissenschaftliche Ausprägung der Netze lag an der Tatsache, dass der Wissenschaftler *Tim Berners Lee*[18], der ab 1989 die Grundlagen des World Wide Web (WWW, oder kurz Web) entwickelte, ursprünglich eine Möglichkeit schaffen wollte, wissenschaftliche Dokumente für alle Projektbeteiligten verfügbar zu machen.

Dies sollte über einfache Textdokumente und die Möglichkeit der Grafikeinbindung realisiert werden. Entscheidend für die weitere Entwicklung war die Idee der Hypertext-Funktionalität, bei der Dokumente Verweise auf andere Dokumente enthalten können, und zwar unabhängig davon, auf welchen Server diese gespeichert waren.[19]

Um eine solche Funktionalität zu realisieren, entwickelte Berners-Lee das Übertragungsprotokoll HTTP, die Dokumentenbeschreibungssprache HTML und das Konzept der URL's.

Da diese Begriffe in engem Zusammenhang mit der zu entwickelnden Software stehen, wird versucht, in den nachfolgenden Kapiteln relevante Basisinformationen zu vermitteln.

[18] Tim Berners-Lee ist Wissenschaftler am Hochenergieforschungszentrum der Europäischen Organisation für Kernforschung (CERN) bei Genf und Mitbegründer der Grundlagen für das World Wide Web

[19] Siehe (**Schmitz et al. 2003,** S. 27)

3.3.1 World Wide Web

Die ursprüngliche Idee von Berners Lee, wissenschaftliche Dokumente zur Verfügung zu stellen, breitete sich im Laufe der Jahre auf eine riesige Anzahl von miteinander verbundenen Dokumenten aus, die keine geordnete Struktur aufweisen. Diese Dokumente können dabei Texte, Bilder, Töne und Filme enthalten, die weltweit auf verschiedenen Rechnern zu Verfügung gestellt werden.

Mit Hilfe von Querverweisen innerhalb dieser Dokumente, so genannten Links oder Hyperlinks, kann wiederum auf andere Dokumente verwiesen werden, woher das „weltweite Spinnennetz“ sein Namen verdankt.

Das World Wide Web (WWW, Web, oder kurz W3) ist einer von vielen im Internet bereitgestellten Diensten, der wie die anderen Dienste auch, auf dem *TCP/IP*[20]-Protokoll beruht.

Eine Übersicht der wichtigsten im Internet genutzten Protokolle und Dienste liefert die Abbildung 7.

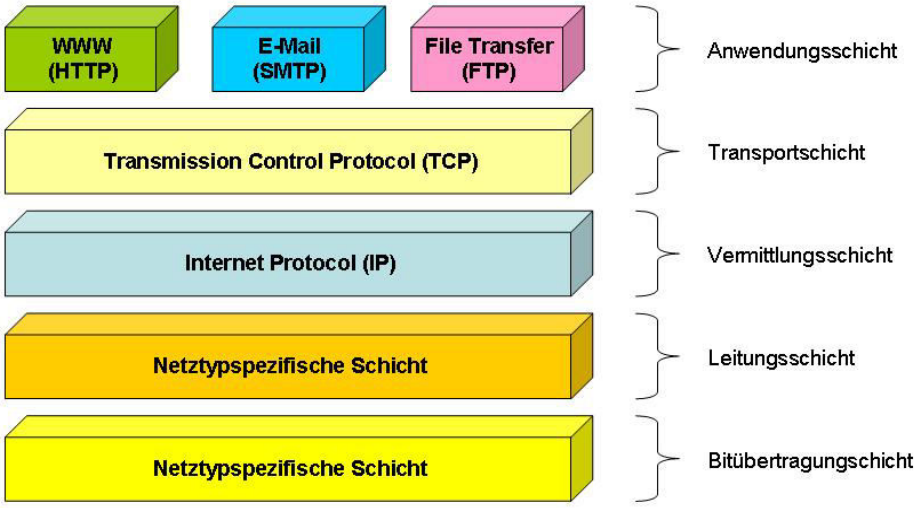


Abbildung 7: Protokolle und Dienste im Internet

[20] TCP/IP ist die Abkürzung für Transmission Control Protocol/Internet Protocol und ist eine Sammlung einiger, sich ergänzender Protokolle aus der so genannten Internet Protocol Suite, die in den Requests For Comments (RFC) beschrieben werden. (siehe z.B. www.rfc-editor.org/ oder www.nic.de/)

Neben dem Dienst Word Wide Web (WWW), welcher auf dem Protokoll HTTP beruht, ist noch als besonders wichtig der Dienst Electronic Mail (e-Mail) anzusehen, der im Internet durch das Protokoll *SMTP*[21] geregelt wird.

Die Weiterentwicklung der im Internet angebotenen Dienste und Funktionalitäten wird durch das *W3-Konsortium (W3C)*[22] vorangetrieben. Hierbei handelt es sich um ein Gremium, welches aus Organisationen und Firmen besteht.

[21] SMTP ist die Abkürzung für Simple Mail Transfer Protocol

[22] Die Arbeit des W3-Konsortiums unterteilt sich in so genannte Aktivitäten (Activities). Aufgrund der Vielzahl der Entwicklungsaktivitäten, wie zum Beispiel in den Bereichen HTML, CSS, XML usw., werden die Einzelaktivitäten in Arbeitsgruppen (Working Groups) und Interessensgruppen (Interest Groups) aufgeteilt. Dabei erfolgt die Ausarbeitung der Inhalte der einzelnen Themen in den Arbeitgruppen. Weitere Informationen über die Aktivitäten des W3-Konsortiums können über den URL: (WWW.w3c.org/) abgerufen werden.

3.3.2 Client-Server-Prinzip

Grundsätzlich handelt es sich beim WWW-Dienst um ein Informationssystem, das auf dem Client-Server-Prinzip beruht. Das heißt, es gibt eine Instanz, die als Kunde (Client) fungiert und Dienste einer anderen Instanz (die des Servers) in Anspruch nimmt.

Realisiert wird die Rolle des Clients im WWW mit Hilfe eines lokalen *Browsers*[23], einem Programm, mit dessen Hilfe es möglich ist WWW-Dokumente zu nutzen.

Die Rolle des Dienstleister, also die des Servers übernimmt ein Programm, das auf einem Rechner im Internet läuft und als *Web-Server*[24] bezeichnet wird.

Die gemeinsame Sprache der beiden Kommunikationspartner ist in diesem Fall das Hypertext Transfer Protocol *(HTTP)*[25].

Eine typische Kommunikation zwischen einem Client (Browser) und einem Server (Web-Server) ist in der nachfolgenden Abbildung 8 darstellt.

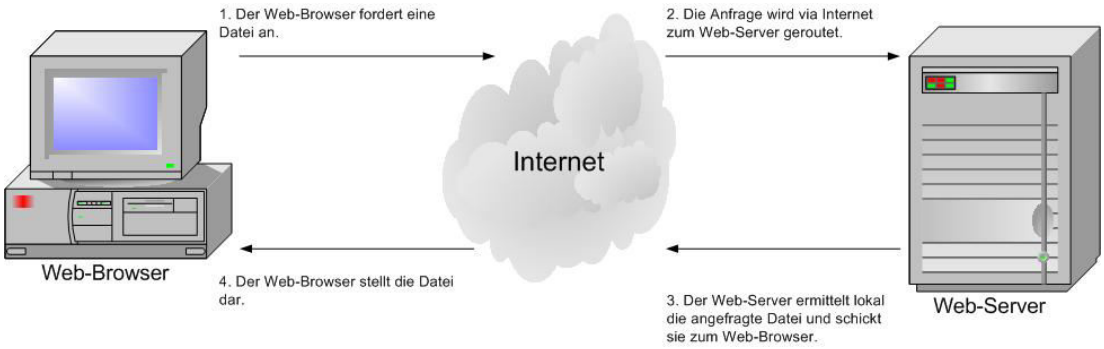


Abbildung 8: Web-Browser und -Server Kommunikation

Das vom Benutzer gewünschte Dokument wird entweder durch Eingabe eines URL's oder durch Anklicken von Verweisen (Hyperlinks) in bereits dargestellten Dokumenten angefordert. Die Anfrage wird mittels der eindeutigen Adressierung über das Internet an den entsprechenden Web-Server gerichtet.

[23] Gängige Browser sind zum Beispiel die Produkte Netscape Navigator der Firma Netscape, der Internet Explorer der Firma Microsoft oder Opera der Firma Opera Software ASA.
[24] Gängige Web-Server sind zum Beispiel der Apache Webserver, der Netscape Enterprise Server der Firma Netscape und der Internet Information Server der Firma Microsoft.
[25] Weitere Informationen über die Spezifikation können in den Request For Comments 2068 unter dem URL: (http://www.ietf.org/rfc/rfc2068.txt; Stand [10.08.2004]) eingeholt werden.

Der Webserver ermittelt dann lokal das angefragte Dokument, wobei es sich nicht nur um HTML-Dokumente handeln muss, sondern es sich ebenso um *Java-Applets*[26], Grafiken oder Audio-Dateien handeln kann.

Das gefundene Dokument wird anschließend an den anfragenden Web-Browser geschickt, der sie dem Benutzer aufbereitet. Sofern weitere Web-Objekte in dem angefragten Dokument vorhanden sind und abhängig davon, welche HTTP-Version vereinbart wurde, wird wiederum eine neue Verbindung aufgebaut oder die bestehende Verbindung erneut genutzt.

3.3.3 Hypertext Markup Language

Die Hypertext Markup Language (HTML) ist neben dem Übertragungsprotokoll HTTP und den Uniform Resource Locators (URL) eine der tragenden Konzepte des Internets.

Einer der Gründe für den Erfolg dürfte die Tatsache sein, dass die HTML im so genannten Klartextformat vorliegt und damit verschiedene Vorteile verbunden sind.

Klartextformat bedeutet, dass die HTML-Dateien mit jedem beliebigen Texteditor bearbeitet werden können, der Daten als Textdateien abspeichern kann. Dies hat den entscheidenden Vorteil, dass die Erstellung von HTML-Dateien nicht an ein kommerzielles Softwareprodukt und überdies nicht an eine bestimmte Plattform gebunden ist. Daher können die erstellten HTML-Dateien auf jedem PC, jeder Workstation, auf jedem Apple Macintosh oder Unix-Rechner dargestellt werden.

Ein weiterer Vorteil des Klartextformats besteht darin, dass es relativ leicht ist dynamisch HTML-Dateien, zum Beispiel bei der Anzeige des Ergebnisses einer Suchanfrage, zu generieren. Von dieser Möglichkeit machen beispielsweise *CGI*[27]-Skripte Gebrauch.[28]

[26] Java Applets sind Java-Programme, die in HTML-Seiten eingebunden werden können. Der Interpreter ist hierbei im Appletviewer bzw. im Browser integriert.
[27] CGI ist die Abkürzung für Common Gateway Interface. Es ermöglicht das Starten von externen Programmen auf dem Webserver und den Datenaustausch mit diesen Programmen.
[28] Vgl. (**Münz 2002**, S. 23)

Bei der HTML handelt sich um eine Sprache, die mit Hilfe der SGML (Standard Generalized Markup Language) definiert wird. SGML ist in der ISO-Norm 8879 definiert. Mittlerweile gibt es eine Erweiterung mit dem Namen *XHTML*[29], auf die an dieser Stelle nicht weiter eingegangen werden kann.

Die HTML ist eine so genannte Auszeichnungssprache, die die Aufgabe hat die logischen Bestandteile wie zum Beispiel Überschriften, Textabsätze, Listen, und Tabellen, eines textorientierten Dokuments auszuzeichnen (= markup) und zu beschreiben.

Dabei werden die den Text beschreibenden Elemente hierarchisch angeordnet und deren Erstreckungsraum mit so genannten Auszeichnungsmarkierungen (Tags) gekennzeichnet.

Die Aufgabe des Browsers ist es, die Elemente innerhalb der HTML-Dateien, die übrigens am Suffix „.html“ oder „.htm“ erkennbar sind, anhand der Tags aufzulösen und dem Benutzer in geeigneter Weise auf dem Bildschirm darzustellen.

Damit diese Visualisierung mit möglichst vielen Browsern verschiedener Softwarehersteller einwandfrei funktioniert, müssen die Browser, die im HTML-Dokument verwendeten Elemente unterstützen. Um dies zu gewährleisten, sollte der verwendete Browser die jeweils aktuelle HTML-Version 4.X unterstützen, die derzeit die geläufige Standard-Version ist.

Die eigentlichen HTML-Elemente werden durch Tags markiert, wobei fast alle HTML-Elemente durch ein einleitendes und ein abschließendes Tag kenntlich gemacht werden und den Gültigkeitsbereich des darzustellenden Elements festlegen. Nachfolgend wird in vier Beispielen auf die Verwendung der Tags eingegangen.

[29] XHTML ist eine Familie aktueller und zukünftiger Dokumenttypen und Module, die HTML 4 nachbilden, eine Untermenge davon bilden bzw. es erweitern. Dokumenttypen aus der XHTML-Familie basieren auf XML und sind letztlich darauf ausgelegt, in Kombination mit auf XML basierenden Benutzerprogrammen eingesetzt zu werden.

1. Beispiel:

```
<h1>Information</h1>
```

In diesem Beispiel wird eine Überschrift 1. Ordnung erzeugt (h= heading= Überschrift). Das einleitende Tag <h1> signalisiert, dass der nachfolgende Text „Information“ in Form einer Überschrift der ersten Ordnung dargestellt werden soll. Das abschließende Tag </h1>, das grundsätzlich mit einem Schrägstrich beginnt, signalisiert das Ende der Überschrift.

Abweichend von dieser Notation gibt es noch HTML-Elemente, die lediglich Stand-alone-Tags benötigen, weil sie keine Information beinhalten.

2. Beispiel:

```
Eine Zeile<br>
Nächste Zeile
```

In diesem Beispiel wird am Ende der ersten Zeile signalisiert, das ein manueller Zeilenumbruch eingefügt werden soll (br= break= Umbruch), ein abschließendes Tag gibt es bei diesem Element nicht.

Weiterhin sei an dieser Stelle noch angemerkt, dass die HTML bei der Verwendung der Elementnamen wie zum Beispiel
 nicht in Groß- und Kleinschreibung unterscheidet und daher
 die gleiche Aussage hat.

Den strukturierten Ansatz erhält die HTML jedoch durch die Möglichkeit Elemente ineinander zu verschachteln.

Im nachfolgenden Beispiel handelt es sich wieder um die gleiche Darstellung wie im ersten Beispiel, mit dem Unterschied, dass zusätzlich noch die Information A zwischen den Tags und kursiv (i= italic= kursiv) dargestellt wird.

3. Beispiel:

```
<h1><i>Information A </i> Information B</h1>
```

Eine Möglichkeit den Elementen weitere Parameter in Form von Attributen zu übergeben wird im letzten Beispiel dargestellt.

4. Beispiel:

```
<h1 align= “center“> Information</h1>
```

Hier wird der Überschrift der 1. Ordnung, kenntlich gemacht durch das Tag <h1> und </h1>, zusätzlich noch die Information übergeben, dass die Überschrift zentriert (align= Ausrichtung und center= zentriert) dargestellt werden soll.[30]

Zum Abschluss dieses Unterkapitels wird an dieser Stelle das Grundgerüst einer HTML-Datei vorgestellt.

Grundsätzlich besteht eine gewöhnliche HTML-Datei aus den folgenden drei Teilen:

1. Dokumenttyp-Angabe (Angabe der verwendeten HTML-Version)
2. Header (Kopfdaten, wie z.B. Angaben zum Titel)
3. Body (Körper – anzuzeigender Inhalt, Text mit Überschriften, Verweisen, Grafikreferenzen usw.)

Quelltext	Beschreibung
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">	Benennung der Document Type Deklaration
<HTML>	Beginn des eigentlichen HTML-Dokuments
<HEAD>	Beginn des Headers
<TITLE> Text des Titels </TITLE>	Text des Titels, der im Rahmen des Browsers angezeigt wird
</HEAD>	Ende des Headers
<BODY bgcolor="#ffffff">	Beginn des Bodies mit Angabe der Hintergrundfarbe
… weiterer Inhalt der Seite …	
<IMG src="bildname.jpg">	Einbindung eines Bildes im JPEG-Format
 	Ein Zeilenumbruch
<HR>	Eine Horizontale Trennlinie (hr= horizontal ruler)
…	
</BODY>	Ende des Bodies
</HTML>	Ende des HTML-Dokuments

Tabelle 1: Grundgerüst einer HTML-Datei

[30] Vgl. (**Münz 2002**, S. 49 ff.)

3.3.4 Uniform Resource Locator

Die Anforderung eines WWW-Dokuments (zum Beispiel eine HTML-Seite) wird durch die Eingabe einer weltweit eindeutigen Adresse, dem Uniform Resource Locator, *(URL)*[31] durchgeführt. Dies geschieht entweder durch explizites Eingeben eines URL's in einem Browser oder durch Anklicken eines Hyperlinks in einem bereits dargestellten Dokument.

Damit das angeforderte Dokument auf dem entsprechenden Rechner (Host) gefunden werden kann, muss eine bestimmte Syntax eingehalten werden. Grundsätzlich besteht ein URL aus sechs Abschnitten, die nachfolgend erläutert werden (siehe Abbildung 9 und Tabelle 2).

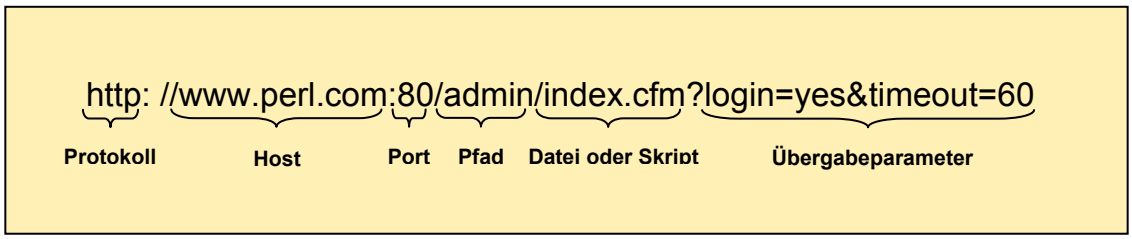


Abbildung 9: Syntax eines WWW-Uniform Resource Locators

Abschnitt	Beschreibung
Protokoll	Das Protokoll mit dessen Hilfe das anforderte Dokument übertragen wird. Dies ist für den WWW-Dienst gewöhnlich das HTTP-Protokoll. Wenn das Protokoll angegeben wird, müssen ein Doppelpunkt und zwei Schrägstriche folgen.
Host	Angabe des Hosts der das Dokument liefert. Entweder spezifiziert durch den DNS-Name oder durch eine IP-Adresse.
Port	Angabe der Portnummer, mit der der Webserver den WWW-Dienst verknüpft hat. Wenn der Default-Port 80 zutrifft, kann die Angabe inklusive des Doppel-punkts entfallen.
Pfad	Pfad zum angeforderten Verzeichnis, Dokument oder zum auszuführenden Skript.
Datei	Dateiname des angeforderten Dokuments
Übergabeparameter	Angabe von optionalen Übergabeparametern, die mit einem Fragezeichen eingeleitet werden müssen, sofern es sich um Suchparameter handelt.

Tabelle 2: Erläuterung der URL-Abschnitte

[31] Weitere Informationen über die Spezifikation eines URL's können in den Request For Comments 1738 unter dem URL: (http://ietf.org/rfc/rfc1738; Stand [10.08.2004]) eingeholt werden.

Aufgrund der Tatsache, dass das Web-Server Programm oftmals nicht das einzige Dienstprogramm auf einem Host ist, muss zusätzlich zur Angabe der IP-Adresse oder des DNS-Namens noch eine Protokollkennung übergeben werden.

Mit Hilfe dieser Angabe ist dem Host eine Unterscheidung des in Anspruch zu nehmenden Dienstes möglich.

Für die im Internet genutzten Standarddienste wurden zu diesem Zweck Portnummern definiert, die als wohlbekannte Portnummern (well-known port numbers) bezeichnet werden und den Nummernbereich von 0 bis 1023 abdecken.

Insgesamt stehen jedem Rechner 65563 Nummern zur Verfügung, wobei der Bereich von 1024 bis 49151 meist zur Kommunikation von Hersteller spezifischen Diensten genutzt wird.

Die Tabelle 3 gibt eine Übersicht über die meist genutzten IP-Port-Nummern:

Port	Dienstname	Request for Comments
20	FTP	959
21	FTP	959
23	Telnet	854
25	SMTP	821
43	Whois	812
53	DNS	1034/1035
70	Gopher	1436
79	Finger	1288
80	http	2068
107	Remote Telnet service	818
110	POP3	1939
119	NNTP	977
143	IMAP2	1064
194	IRC	1459
220	IMAP4	2060
389	LDAP	2251
443	HTTPS	2818
540	UUCP	976
1723	PPTP	2637

Tabelle 3: IP-Portnummern

3.4 Das Virtuelle Rathaus der Stadt Hagen

3.4.1 Historie

Die zunehmende Nutzung des Internets durch die privaten Haushalte und das damit verbundene Potential für Dienstleistungsorientierung, Bürgerbeteiligung, Produktivität und Wirtschaftlichkeit im öffentlichen Sektor[32], veranlasste die Bundesregierung im September 2002 eine Initiative mit dem Namen *„E-Government BundOnline 2005“*[33] ins Leben zu rufen.

Ziel dieser Initiative war die Bereitstellung von Dienstleistungen der öffentlichen Hand mit Hilfe des Internets. Dem Bürger, der Wirtschaft, der Wissenschaft und der Verwaltung selbst sollte es möglich sein, Dienstleistungen, einfacher, schneller und kostengünstiger nutzen zu können.

Bereits einige Jahre vor der Initiative der Bundesregierung entwickelte der Hagener Betrieb für Informationstechnologie und die Kooperationspartner im Ennepe-Ruhr-Kreis das eGovernment-Portal - mit dem Namen „Virtuelles Rathaus“.

Im Jahr 1997 fand der Start dieses Internetprojekts statt, so dass der HABIT in Zusammenarbeit mit den Hauptkooperationspartnern Fernuniversität Hagen und der i-World GmbH seit dem Jahre 2002 eines der funktional am weitesten fortgeschrittenen eGovernment-Portale in Deutschland anbieten konnte. Dies zeigt sich im Erwerb verschiedener Preise und Auszeichnungen, worunter zum Beispiel eine Auszeichnung des Speyerer Qualitätswettbewerbs der deutschen Hochschule für Verwaltungswissenschaften zu nennen ist.

Die Bereitstellung dieses Internetportals ermöglicht es den Bürgern der Stadt Hagen ihr Rathaus fast rund um die Uhr zu nutzen und Dienstleistungen in Anspruch zu nehmen.

[32] Vgl. (**Lucke 2000**, S. 1)
[33] Aktuelle Informationen zum Fortschritt und weitere Detailinformationen der Initiative können unter dem URL: (http://www.bund.de/BundOnline-2005-.6164.htm; [Stand: 10.08.2004]) abgerufen werden.

3.4.2 Produkte und Dienstleistungen

Die Produktpalette des Virtuellen Rathauses besteht aus Angeboten, die sich im Grad der Komplexität hinsichtlich der Kommunikation zwischen dem Bürger und dem System stark unterscheiden. Angefangen bei der reinen Informationsbereitstellung - über die Kommunikation und Interaktion bis hin zu Transaktionen der Bürger bei der Nutzung von kostenpflichtigen Dienstleistungen - wird ein breites Spektrum angeboten (siehe Abbildung 10).

Abbildung 10: Internetportal des Virtuellen Rathauses der Stadt Hagen
(URL: http://vrhagen.stadt-hagen.de/; [Stand 05.05.2004])

Zu den Kernfunktionen des Virtuellen Rathauses können folgende Dienstleistungen bzw. Produkte zusammengefasst werden:

- Verzeichnis aller Verwaltungsdienstleistungen
- Bereitstellung von Verwaltungsinformationen zu allen städtischen Dienstleistungen
- Bereitstellung von Formularen und städtischen Rechtsvorschriften
- Kommunikation mit allen Dienststellen der Verwaltung via elektronischer Post
- Bereitstellung von kostenpflichtigen Produkten mittels verschiedener Online-Bezahlverfahren

Bei der Bereitstellung der Dienstleistungen und Produkte ist zu beachten, dass zwischen "freien Produkten" und „anmeldepflichtigen Produkten“ unterschieden werden muss.

Dienstleistungen die allen Besuchern des Virtuellen Rathauses zugänglich sind, werden in diesem Zusammenhang als "freie Produkte" bezeichnet.

Produkte, welche eine eindeutige Authentifizierung der abrufenden Person erfordern, als „anmeldepflichtige Produkte“.

Hierzu gehören zum Beispiel:

"Freies Produkt":

- im Bereich Fahrzeuge und Verkehr (Zulassungswesen): die Reservierung eines Wunschkennzeichens
- die Bereitstellung von diversen Formularen und Anträgen aus verschiedensten Bereichen (Rat und Verwaltung, Einwohner und Lebenslagen, etc.)

"Anmeldepflichtiges Produkt":

- im Bereich Planen und Bauen (Geoinformationen und Liegenschaftskataster): Angabe des Eigentümers eines Grundstücks anhand des Flurstückskennzeichens

3.4.3 Architektur des Virtuellen Rathauses

Die Systemarchitektur des Virtuellen Rathauses ist vorrangig geprägt durch einen Verbund von Servern, die auf dem Windows Betriebsystem basieren. Dieser Verbund von Servern wird mit Hilfe einer auf den TCP/IP-Standard basierenden Netzinfrastruktur realisiert.

Die folgende Abbildung 11 liefert einen Überblick über die Systemarchitektur:

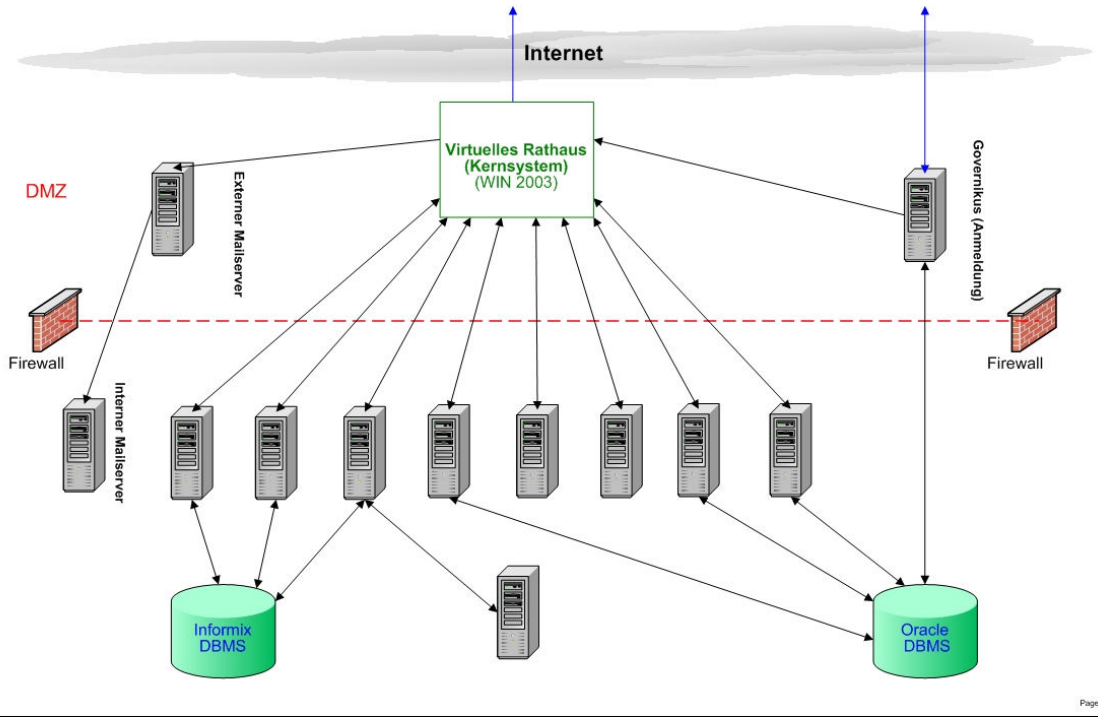


Abbildung 11: Serverbeteiligung am Virtuellen Rathaus

Das Kernsystem dieses Serververbundes bildet ein *ColdFusion Application Server MX® (Version 6.1)*[34] in Verbindung mit einem *Internet Information Server® (Version 6.0)*[35], das an verschiedene weitere Server angebunden ist.

Diese Server stellen die angefragten Informationen entweder selbst zur Verfügung oder mit Hilfe des Zugriffs auf weitere Fachverfahren, die ihrerseits auf Datenbanken zugreifen.

[34] ColdFusion Application Server MX Version 6.1 ist die Produktbezeichnung der Firma Macromedia für einen Web-Applikations-Server.
[35] Internet Information Server (IIS) ist die Produktbezeichnung eines Web-Servers der Firma Microsoft.

Bei den eingesetzten Datenbanken handelt es sich um zwei Datenbankserver, basierend auf den Datenbankmanagementsystemen der Firmen Informix® und Oracle®. Die E-Mail-Kommunikation wird mit einer Kombination aus einem ColdFusion Application Server, einem externen und einem internen Mailserver realisiert.

Die Bereitstellung von anmeldepflichtigen Dienstleistungen die eine Authentifizierung des Benutzers erfordern, werden mit dem Produkt Governikus® der Firma Bremen Onlines Services GmbH & Co. KG abgewickelt. Hierbei handelt es sich um eine modulare *Middleware*[36], die die sichere und signierte Übermittlung von rechtsverbindlichen Erklärungen ermöglicht. Governikus nutzt hierfür die Internetkommunikationsstandards *XML* und *OSCI*[37]. Um den Sicherheitsaspekt einer Systemarchitektur einer öffentlichen Verwaltung, die an das weltweite Internet angeschlossen ist, in höchstem Maße Rechnung zu tragen, wurde mit einer Firewall-Technologie eine Demilitarisierte Zone (*DMZ*[38]) geschaffen.

Die Abbildung 12 konkretisiert noch mal die technische Struktur des Kernsystems:

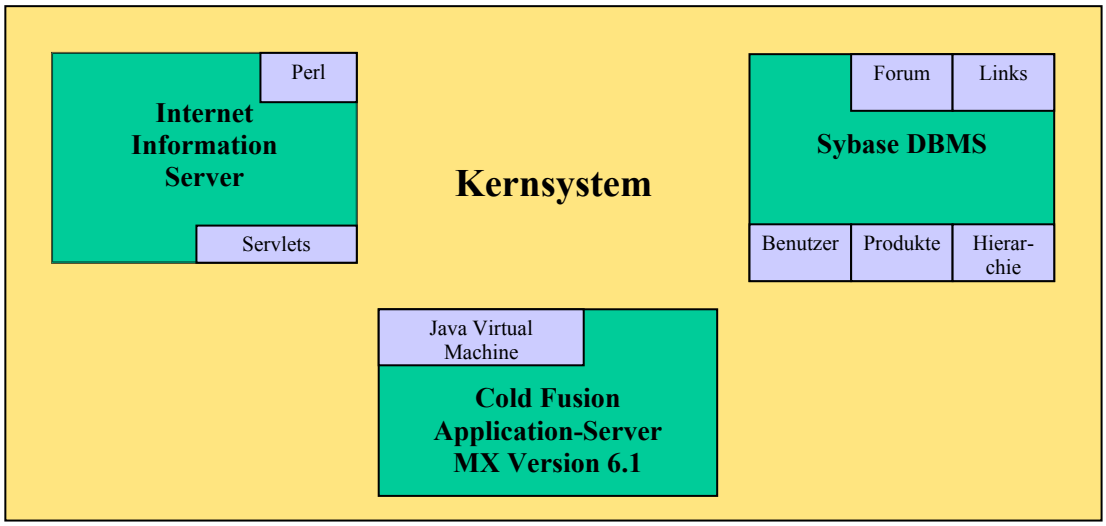


Abbildung 12: Technische Struktur des Kernsystems

[36] Unter Middleware versteht man eine Software, die Informationen zwischen mehreren, verteilten und verschiedenen Programmen (Anwendungen) zuverlässig verteilt und die Kommunikation und den Datentransfer steuert und kontrolliert.
[37] OSCI steht für Online Services Computer Interface. Hierbei handelt es sich um einen Standard, der mit dem Protokoll OSCI-Transport die sichere Übertragung von elektronisch signierten Nachrichten ermöglicht.
[38] Bei einer DMZ ("Demilitarized Zone"/"entmilitarisierte Zone") handelt es sich um einen geschützten Rechnerverbund, der sich zwischen 2 Netzwerken befindet. Der Rechnerverbund wird jeweils durch einen Paketfilter gegen das dahinter stehende Netz abgeschirmt.
URL: (http://www.net-lexikon.de/DMZ.html [Stand: 15.06.2004])

3.4.4 ColdFusion Applikations-Server

Der ColdFusion Applikationsserver bildet in Verbindung mit dem Webserver der Firma Microsoft das Kernsystem des Virtuellen Rathauses. Es stellt sich die Frage, weshalb neben dem Web-Server, der sowieso schon Anfragen des Web-Browsers beantwortet, noch einen weiteren Applikationsserver benötigt wird.

Der Webserver ist ein ständig laufendes Programm, das ununterbrochen auf Anfragen eines Clients wartet. Empfängt der Webserver eine solche Anfrage, sucht er auf dem lokalen Speichermedium nach den passenden Dokumenten und schickt diese zurück zum Client (Web-Browser). Dabei kommt dem Webserver die statuslose Kommunikation im Web zugute. Das heißt, wenn der Client sein angefordertes Dokument erhalten hat, ist die Arbeit für den Webserver zumindest für diese Verbindung erledigt. Es werden keine Informationen über die Verbindung gespeichert und keine offenen Informationskanäle beibehalten.

Das Problem bei dieser Art der Kommunikation ist, dass diese sich nur für die Bereitstellung von statischen Inhalten und Informationen eignet. Um den Anforderungen eines eGovernment-Portals zu genügen, das heißt um dynamisch auf die Anfragen der Bürger zu antworten bzw. mit ihnen in Interaktion treten zu können, müssen andere Mechanismen genutzt werden.

Eine Möglichkeit dem Problem der statischen Bereitstellung von Web-Inhalten zu begegnen, ist die Nutzung eines Web-Applikations-Servers. Im Falle des Virtuellen Rathauses wird das Produkt ColdFusion MX der Firma Macromedia genutzt.

Hierbei handelt es sich um ein Programm, welches die Anfragen des Clients über den Web-Server entgegen nimmt, die Antwort um weitere Daten ergänzt und wieder über den Web-Server beantwortet.

Das heißt, die HTML-Seiten werden erst zur Laufzeit generiert. Daher wird in diesem Zusammenhang von dynamisch erzeugten Webseiten gesprochen. Die Darstellung erweckt beim Benutzer des Browsers den Eindruck, als hätte er eine statische Webseite aufgerufen. Die nachfolgende Abbildung 13 veranschaulicht das Zusammenspiel zwischen einem Applikationsserver und einem Web-Server und die Bereitstellung der dynamischen Webseiten.

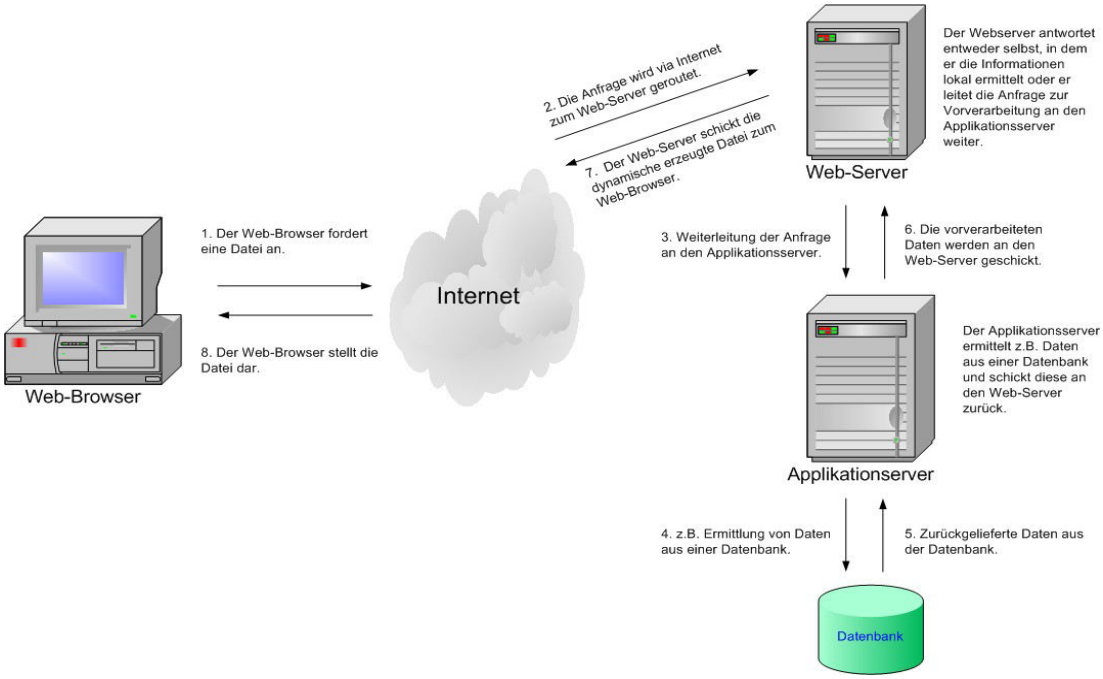


Abbildung 13: Interaktion mit einem Applikationsserver

Wenn der Webserver von einem Web-Browser eine Anfrage über das Internet erhält, ermittelt dieser, ob es sich um die Bereitstellung einer statischen Web-Seite handelt, die er in diesem Fall selbst zur Verfügung stellen kann oder um die Bereitstellung von Informationen, die eine Interaktion mit dem Applikationsserver erfordern. Diese Entscheidung trifft der Web-Server anhand des Multipurpose Internet Mail Ex*tension*[39] Typs (MIME) bzw. an der Datei-Erweiterung.

Wenn der MIME-Typ anzeigt, dass es sich um eine einfache Web-Seite handelt (z.B. an der HTM-Endung), ermittelt der Web-Server lokal die Datei und schickt sie unverändert zurück an den anfragenden Web-Browser. Falls anhand des MIME-Typs erkennbar ist, dass es sich bei dem angefragten Dokument um eine Datei handelt die vom Applikationsserver bearbeitet werden soll (z.B. an der CFM-Endung), wird die Anfrage an den Applikationsserver weitergeleitet. Dieser ermittelt die spezifischen Informationen, z.B. aus einer Datenbank, und liefert die gesamten Daten an den Web-Server, der diese zu einem HTML-Dokument zusammenfügt.

[39] MIME legt die Struktur und den Aufbau von E-Mails und anderer Internetnachrichten fest. Es sind mehrere Kodierungsmethoden spezifiziert, die die Übertragung von Sonderzeichen in Texten sowie Dateien von Nicht-Text-Dokumenten, wie Bilder, Audio und Video in textbasierten Übertragungssystemen, wie E-Mail oder Usenet, ermöglichen. Die Nicht-Text-Elemente werden beim Versender kodiert und beim Empfänger wieder dekodiert.
Vgl. URL: (http://de.wikipedia.org/wiki/MIME; [Stand: 03.09.04])

3.5 Perl

In diesem Kapitel wird erläutert, warum bei der Wahl der Implementierungsprache zur Umsetzung der Anwendungssoftware, die Programmiersprache Sprache Perl den Vorzug bekam. Dabei werden speziell die Aspekte der Programmiersprache hervorgehoben, die sich als besonders hilfreich bei der Implementierung herausstellten.

Was ist Perl?

Perl ist die Abkürzung für "practical extraction and report language".

Anhand dieser Bezeichnung lässt sich schon die Intention des Erfinders Larry Wall erkennen, eine Programmiersprache zu entwickeln, deren Hauptaugenmerk auf die einfache Erstellung von effizienten Programmen lag.

Um dieses Ziel zu erreichen, bediente er sich nützlicher Merkmale der Compilersprache C und verschiedener Hilfsmittel unter Unix, wie zum Beispiel dem Stream- Editor *sed*[40] und der Programmiersprache *awk*.[41]

Im Gegensatz zu den Compilersprachen wie zum Beispiel C oder C++ handelt es sich bei der Programmiersprache Perl um eine Skriptsprache.

Das heißt, die Programme werden als ASCII-Files gespeichert und erst unmittelbar vor der Ausführung interpretiert. Daher sind Perl-Programme leicht editierbar und auf andere Rechnersysteme ohne Probleme übertragbar.

Die folgende Auflistung zeigt einen Ausschnitt der möglichen Einsatzgebiete von Perl:

- CGI-Programmierung
- Systemverwaltung
- Anwendungsprogrammierung
- Datenbankverwaltung

Die wichtigsten Vorteile, die den Autor dazu bewogen die Anwendungssoftware mit Perl umzusetzen, werden nachfolgend erläutert.

---

[40] sed ist die Abkürzung für Stream-Editor. Hierbei handelt es sich um eine UNIX Applikation, welche eine zeilenweise Bearbeitung von Textdateien ermöglicht.

[41] awk ist eine UNIX Applikation, die nach ihren Urhebern **A**lfred V. Aho, Peter J. **W**einberger und Brian W. **K**ernighan benannt wurde. Sie dient vorrangig zur zeilenweisen Bearbeitung von Textfiles, wobei auch eine Interaktion mit dem Benutzer möglich ist.

3.5.1 Vorteile von Perl

1. Kostenlose Verfügbarkeit:

Perl ist eine frei verfügbare Sprache (inklusive des Quellcodes), die der „Artistic Licence“ und der „General Public Licence“ unterliegt. Durch die Verwendung der Programmiersprache Perl wurde damit der Anforderung des Auftraggebers nachgekommen, keine Kosten zu verursachen.

2. Perl ist eine interpretierte Sprache:

Perl ist eine Interpretersprache. Der Vorteil liegt speziell darin, dass durch den Wegfall der Kompilierungszeit sich ein kürzerer Entwicklungszyklus ergibt.

3. Perl ist multiplattformfähig:

Das heißt, Perl ist für alle gängigen Betriebssysteme verfügbar, wobei die ursprüngliche Herkunft beim Betriebssystem Unix liegt. Zu den ebenfalls unterstützen Betriebssystemen zählen zum Beispiel Microsoft Windows, Linux, Macintosh, AS/400-Großrechner und MVS-(OS/390).

Da die Systemarchitektur des Virtuellen Rathauses vorrangig aus Rechnern besteht, die auf dem Windows Betriebssystem beruhen, ist ebenfalls in diesem Punkt die Vorgabe des HABIT erfüllt worden.

Sofern in Zukunft eine Portierung auf ein anderes Betriebssystem notwendig wäre, würde das keine Probleme verursachen, wenn bei der Implementierung des Programms darauf geachtet wird, dass keine Betriebssystem abhängigen Module verwendet werden.

Bei der Erstellung von VirTest wird auf die Perl Version 5.6.1. (Build 638) von *ActiveState*[42] zurückgegriffen. Hierbei handelt es sich um eine Perl Distribution, die für das Windows Betriebssystem entwickelt wurde.

[42] Die Firma ActiveState bietet diverse Skriptsprachen, wie zum Beispiel Perl, PHP, TCL, Python und die entsprechenden Entwicklungs-Tools an. URL: (http://www.activestate.com/)

4. Umfangreiches Modul-Archiv:

Perl steht ein umfangreiches *CPAN-Archiv*[43] zur Verfügung, welches ebenfalls frei verfügbar ist, und eine Reihe von fertigen Modulen anbietet, die es ermöglichen bestimmte Aufgabenstellungen effizient zu erledigen.

Hierzu zählt zum Beispiel das Modul mit der Bezeichnung TK, ohne das es nicht möglich gewesen wäre, ein Graphical User Interface zu realisieren.

Weitere Beispiele sind das Modul DBI, mit dem die Datenbankanbindung realisiert werden kann und das Modul LWP, mit dem es möglicht wird, eine Kommunikation mit dem Internet herzustellen.

In dem nachfolgenden Quellcode-Auszug wird dargestellt, wie Module in einem Perl-Script eingebunden werden können.

```
# Das sharp-Zeichen leitet einen Kommentar ein.

###### Eingebundene Module
…

use strict;       # Variablendeklaration erzwingen
use warnings;     # Einschalten von Warnungen
use Tk;           # Toolkit zur Erstellung des GUI
use DBI;          # Modul zur Datenbankanbindung einbinden
use Net::SMTP;    # Modul für die e-Mail-Kommunikation einbinden
…
```

Abbildung 14: Einbindung von Modulen in Perl-Skripte

Es wird noch zu einem späteren Zeitpunkt im Kapitel Systemarchitektur und Implementierung näher auf die Nutzung der wichtigsten Module eingegangen.

[43] CPAN ist die Abkürzung für Comprehensive Perl Archive Network. Es handelt sich hierbei um einen Verbund von gespiegelten FTP-Servern, die unter dem URL [http:www.perl.com/CPAN.CPAN.html] nützliche Module für die Integration in Perl Programme zur Verfügung stellen.

5. Mächtigkeit:

Die Mächtigkeit, also der Umfang der Möglichkeiten, der Programmiersprache Perl beruhen zum einen auf die große Vielfalt der angebotenen fertigen Module, zum anderen auf die Integration von nützlichen Applikationen und Merkmalen aus dem UNIX Umfeld.

Ein besonders wichtiges Merkmal der Programmiersprache Perl ist die Möglichkeit reguläre Ausdrücke nutzen zu können.

Was ist ein regulärer Ausdruck?

„Ein regulärer Ausdruck ist eine äußerst kompakte Beschreibung eines Textmusters, genauer gesagt einer ganzen Klasse von möglichen Texten. Mit Hilfe dieser Ausdrücke, lassen sich dann für Strings typische Operationen, wie Suchen beziehungsweise Suchen und Ersetzen generisch angeben.“[44]

Da es bei einigen Funktionen der Anwendungssoftware VirTest notwendig ist, HTML-Seiten nach bestimmten Inhalten zu durchsuchen, ist dies eine wichtige Eigenschaft.

Weitere Anwendungsmöglichkeiten der regulären Ausdrücke zeigt die folgende Auflistung:

- Eingabe-Validierung - es wird sichergestellt, dass der Benutzer die gewünschten Daten eingegeben hat.
- Herausziehen bestimmter Teile einer Datei, die einem bestimmten Kriterium entsprechen.
- Zerlegung eines Strings auf der Basis bestimmter Trennzeichen-Felder.
- Das Zählen der Vorkommen eines Musters in einem String.
- Suchen- und Ersetzen-Operationen - einen String suchen, der mit einem vorgegebenen Muster übereinstimmt, und ihn durch einen anderen String ersetzen

[44] Siehe (**Hajji, 2000**, S. 82)

6. Effizienz

Obwohl Perl eine Skriptsprache ist und interpretiert wird, weist Perl dennoch eine hohe Ausführungsgeschwindigkeit auf. Dies liegt an der Tatsache, dass Perl vor der Ausführung vom Interpreter in einen internen Bytecode übersetzt und optimiert wird.

7. Objektorientierte Features

Perl bietet seit der Version 5 die Möglichkeit Programme objektorientiert zu implementieren. Dabei ist zu beachten, dass nicht alle Methoden und Prinzipien der Objektorientierung vollständig unterstützt werden beziehungsweise nicht alle Methoden ohne einen Mehraufwand umgesetzt werden können.

Dazu zählen zum Beispiel:

- Perl stellt keinen Automatismus zur Instanzierung von Objekten bereit
- Abgeleitete Objekte erben nicht die Attribute der Basisklasse

Daher wurde auf eine objektorientierte Implementierung der Anwendung verzichtet.

Da einige Module des CPAN-Archives, die bei der Umsetzung der Diplomarbeit genutzt wurden, eine objektorientierte Schnittstelle aufweisen, können Begriffe wie Klasse, Objekt oder Methoden speziell in den Kapiteln auftauchen, in denen diese Module beschrieben werden.

3.5.2 Nachteile von Perl

Wie jede andere Programmiersprache, ist Perl ebenfalls mit Nachteilen behaftet, von denen an dieser Stelle zwei erläutert werden:

1. Perl-Interpreter auf dem Zielrechner erforderlich:

Die Vorteile einer Interpretersprache werden über die Tatsache „erkauft“, dass der Perl-Interpreter auf dem ausführenden Zielsystem installiert sein muss.

Das heißt im Falle einer geplanten Distribution muss sowohl der Interpreter als auch die verwendeten Module mitgeliefert werden.

In diesem Fall handelt es sich um eine Applikation, die lediglich auf einem einzelnen Rechner installiert werden soll, so dass dieser Umstand kein Problem darstellt.

2. Kryptische Programme:

Die effiziente Erstellung von Programmen ist gekoppelt mit der Möglichkeit Programme zu schreiben, die unübersichtlich und nicht verständlich wirken.

Dies liegt zum einen daran, dass es möglich ist, viele Sonderzeichen einzusetzen, die abhängig vom Kontext eine andere Funktion ausführen. Zum anderen ermöglicht Perl die Nutzung von komplexen Datenstrukturen in einer äußert kurzen Schreibweise, so dass die Programme schlecht „lesbar“ erscheinen.

3.6 Datenbank MySQL

Um die Testparameter für die Testdurchführung innerhalb von VirTest nicht nur persistent, sondern möglichst *konsistent*[45] und *redundanzfrei*[46] speichern zu können, ist eine Speicherung der Daten in eine Datenbank unumgänglich. Nach Faeskorn-Woyke versteht man unter einer Datenbank *„eine Ansammlung von Daten, die allen Benutzern bzw. Anwendungen zur Verfügung steht und in der die Daten nach einheitlichen Regeln abgespeichert werden. Eine Datenbank besteht aus einer Datenbasis und einem Datenbasisverwaltungssystem (DBMS)“*[47].

Die folgende Abbildung 15 zeigt den generellen Aufbau und die Schnittstellen zu einer Datenbank.

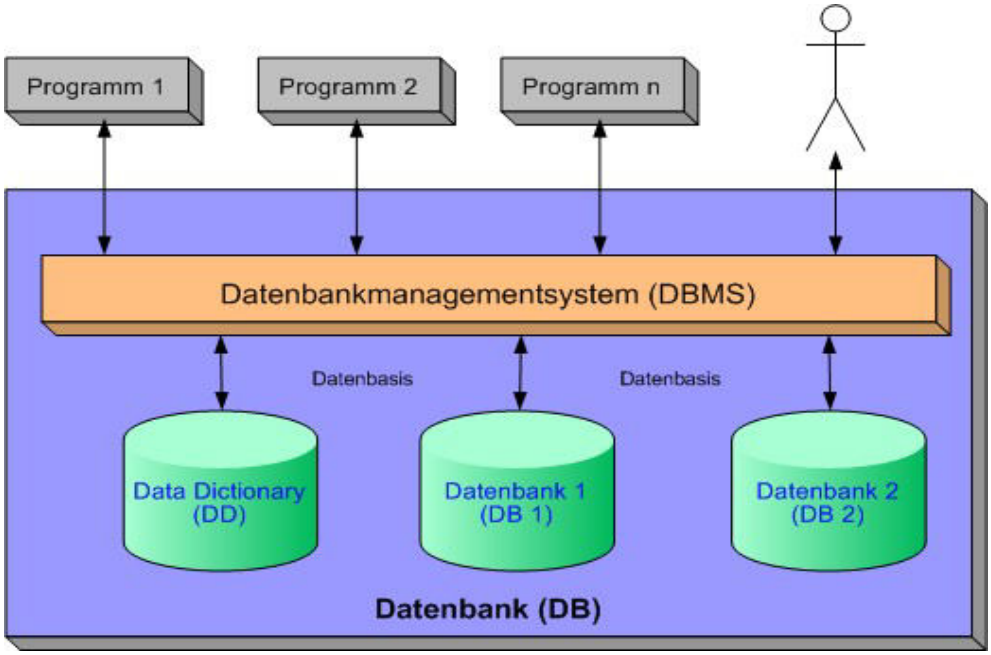


Abbildung 15: Bestandteile und Schnittstellen einer Datenbank

[45] Unter Konsistenz einer Datenbank versteht man die widerspruchsfreie Speicherung der Daten. Zu einem inkonsistenten Zustand kann es z.B. kommen, wenn aufgrund zweier Änderungen ein Attribut in verschiedenen Dateien unterschiedliche Werte hat. Vgl. (**Faeskorn**, 2001, S. 8)

[46] Unter Redundanz versteht man die Mehrfachspeicherung von Daten in unterschiedlichen Dateien. Dieser Zustand tritt häufig ein, wenn die entsprechenden Daten nicht zentral erfasst und verarbeitet werden. Vgl. (**Faeskorn**, 2001, S. 8)

[47] „Das Data Base Management System (DBMS) ist ein Softwaresystem zur Verwaltung der Datenbasis. Alle Prozesse der Datenbank werden gesteuert und verschiedene Dienstleistungen zur Verfügung gestellt. Durch eigene Parametrisierung wird das System an spezielle Bedürfnisse angepasst. Dazu gehören z.B. Funktionen zum Speichern der Daten, zur Verwaltung der Ressourcen und Maßnahmen zur Sicherung der Datenkonsistenz. Ein DBMS liegt in der Funktionalität zwischen Anwendungsprogramm und Betriebssystem.“ Siehe (**Faeskorn**, 2001, S. 11)

Aufgrund der Vorgabe seitens des Auftraggebers, keine Kosten zu verursachen, kann nicht auf eine kommerzielle Datenbank, wie zum Beispiel eine der Firmen wie Oracle oder Sybase zurückgegriffen werden. Zur Auswahl stehen daher nur Open Source Datenbanken, wie *PostgreSQL*[48] oder *MySQL*™[49].

Obwohl die Datenbank PostgreSQL eine größere Funktionsvielfalt anbieten kann als das Produkt MySQL, fiel die Entscheidung auf die Datenbank MySQL (Version 4.0.20d).

Die Gründe für diese Entscheidung werden nachfolgend erläutert:

- MySQL und Perl stellen in einer Vielzahl von Web-Anwendungen eine bewährte Kombination dar. Dadurch kann von einer ausgereiften Schnittstelle (Application Programming Interface, API) zwischen der Datenbank und der Programmiersprache ausgegangen werden.
- Die Installation, speziell unter dem Betriebssystem Windows, ist sehr einfach. Dieser Punkt ist wichtig, um eine entsprechende Entwicklungsumgebung zu schaffen. Zum anderen sind bei der Integration der gesamten Anwendungssoftware in den Produktionsbetrieb, inklusive Datenbank, keine bzw. nur geringe Probleme zu erwarten.
- Es existiert eine sehr umfangreiche Auswahl an Literatur, speziell was die Schnittstelle (API) zur Programmiersprache Perl betrifft. Daher kann bei der Implementierung auf einen relativ großen Erfahrungsschatz zurückgegriffen werden und auftretende Problemstellungen gegebenenfalls schnell gelöst werden.

[48] PostgreSQL ist ein objekt-relationales Datenbank-Managementsystem (DMBS), dessen Source-Code und Dokumentation unter der BSD-Lizenz frei erhältlich sind und an dem von zahlreichen Entwicklern weltweit gearbeitet wird.

[49] Das Produkt MySQL ist ein relationales Datenbank-Managementsystem (DMBS) und wird von der Firma MySQL AB, mit Sitz in Uppsalla (Schweden) unter einer Doppellizenz weiterentwickelt. Daher ist es möglich MySQL zum einen frei unter der GNU General Public Licence (GPL) oder kommerziell zu erwerben.

- MySQL hat speziell beim Lesevorgang von kleinen Tabellen eine höhere Performance als PostgreSQL. Durch den Verzicht auf verschiedene Funktionen, wie zum Beispiel Stored Procedures, Views, Trigger und Unterabfragen, wurde die Datenbank von den Entwicklern bewusst „schlank“ gehalten und somit leistungsfähiger. Da die Testdurchführung der Anwendungssoftware abhängig ist von der zeitnahen Bereitstellung der Testparameter aus der Datenbank, sollte in diesem Fall eine schnelle Datenbank gewählt werden.
- Mit dem *CASE*[50]-Tool DBDesigner 4, welches ebenfalls unter der GNU General Public License frei verfügbar ist, steht ein Hilfsmittel zur Verfügung, mit dem es möglich ist, das erstellte Datenmodell direkt in die MySQL-Datenbank zu integrieren. Weiterhin ist es mit diesem Entwicklungstool leicht möglich Testdaten zu administrieren, um damit die Datenbankschnittstelle während der Implementierung zu testen.

[50] CASE ist die Abkürzung für Computer Aided Software Engineering. Dementsprechend handelt es sich hierbei um eine Software, mit deren Hilfe computergestützt Softwareentwicklung betrieben werden kann.

3.6.1 Merkmale von MySQL

Die nachfolgende Liste enthält wichtige Charakteristika der Datenbank MySQL, die für die Implementierung von VirTest relevant waren.

Eine aktuelle Liste aller Merkmale der Datenbank MySQL ist unter dem URL: http://dev.mysql.com/doc/mysql/de/features.html (Stand: 11.03.05) zu finden.

- Diverse Application Programm Interfaces (API's für die Programmiersprachen C-, C++-, Eiffel-, Java-, Perl-, PHP-, Python- und Tcl).
- Unterstützung von mehr als 20 Betriebssystemplattformen (z.B. AIX, Amiga, HP-UX, Linux, Mac OS, OS/2 Warp, Solaris 2.5, Windows 95/98/NT/2000)
- Unterstützung der gängigsten Datentypen (z.B. INTEGER, FLOAT, DOUBLE, CHAR, VARCHAR, TEXT, BLOB, DATE, TIME, DATETIME)
- Unterstützung von Operatoren und Funktionen in SELECT- und WHERE-Teilen von Anfragen.
- Unterstützung für SQL-GROUP BY und ORDER BY- Klauseln
- Unterstützung von Datensätzen fester und variabler Länge.
- Unterstützung von Datenbanken, die bis zu 50 Mio. Datensätze enthalten
- Alle Spalten können Vorgabewerte (Defaults) haben.
- Unterstützung von Alias-Namen auf Tabellen und Spalten (gemäß SQL92-Standard).
- DELETE, INSERT, REPLACE und UPDATE geben die Anzahl der Zeilen zurück, die geändert wurden (bzw. betroffen sind).
- Der MySQL-spezifische SHOW-Befehl kann benutzt werden, um Informationen über Datenbanken, Tabellen und Indexe zu erhalten.

4 Entwurfsphase

4.1 Modellierungssichten

Um eine neue Anwendungssoftware „vollständig“ zu modellieren, sind verschiedene Gesichtspunkte und ihre Zusammenhänge zu beschreiben. Dazu zählen die Funktionen der Anwendungssoftware, die Daten und das Verhalten.

Die Abbildung 16 verdeutlicht die drei wichtigsten Modellierungssichten:

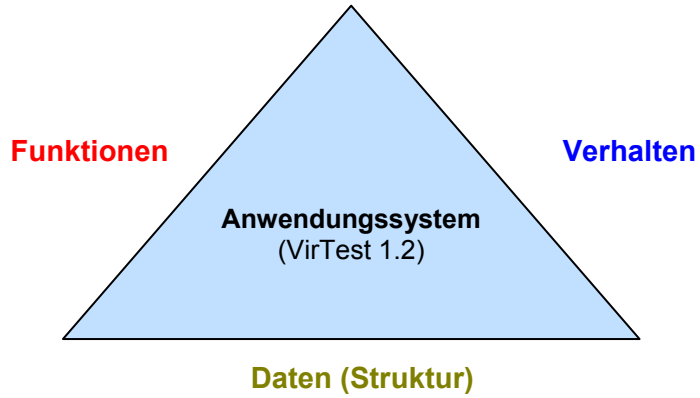


Abbildung 16: Modellierungssichten

Ziel der Modellierung ist es den Grad der Komplexität zu verringern, in dem die Problemstellung abstrahiert, strukturiert und klassifiziert wird.[51] Hinsichtlich der verschiedenen Sichten gibt es mehr oder weniger geeignete Konzepte, die Modellierung durchzuführen. Auch die Erfahrung im Umgang mit den verschiedenen Konzepten spielt bei der Auswahl des geeigneten Konzepts eine wichtige Rolle.

Um die unterschiedlichen Gesichtspunkte zu modellieren, wurde bei der Erstellung dieser Arbeit auf eine Mischung aus bewährten Basiskonzepten, wie der Entity-Relationship-Modellierung (ER-Modellierung), und neuen Konzepten aus der Unified-Modeling-Language (UML), dem Klassen- und dem Use Case-Diagramm zurückgegriffen. Der Grund für die Mischung der Modellierungskonzepte war, dass die Zielsprache Perl nicht alle Konzepte der Objektorientierung unterstützt und daher eine vollständig objektorientierte Modellierung in diesem Fall nicht sinnvoll erschien und nicht möglich war.

[51] Vgl. (**Herde**, 2001**,** S. 2)

4.2 Klassendiagram von VirTest 1.2

Bei der Unified Modeling Language *(UML)*[52] handelt es sich um eine Modellierungssprache, die sich zum Standard im Bereich der objektorientierten Softwareentwicklung entwickelt hat.

Das Klassendiagramm ist dabei eine von mehreren möglichen Notationen aus der Unified Modeling Language, mit der es möglich ist, die logische Struktur der Software zu modellieren.

Das Klassendiagramm beantwortet die Fragen:

- Welche Dinge muss das Anwendungssystem kennen?
- Mit welchen Daten soll das Anwendungssystem umgehen?
- In welchem Verhältnis stehen die Klassen zueinander?
- Welche Funktionen kann das Anwendungssystem ausführen?

Neben dem Klassendiagramm gibt es noch die nachfolgend aufgeführten Diagramme, die je nach Zielsetzung des zu modellierenden Gesichtspunktes Anwendung finden.

- Use-Case Diagramm
- Sequenzdiagramm
- Kollaborationsdiagramm
- Zustandsdiagramm
- Aktivitätsdiagramm

[52] Die UML ist eine grafische Notation für die Erstellung objektorientierter Modelle, die unabhängig von einer Programmiersprache dargestellt werden können. Die Version UML 2.0 ist die derzeit aktuellste Version.

Im Klassendiagramm werden die Klassensymbole mit verschiedenen weiteren Symbolen - wie zum Beispiel der Assoziation oder der Vererbung - eingetragen und zueinander in Beziehung gesetzt.

Was ist eine Klasse?

Nach Balzert[53] definiert eine Klasse „*für eine Kollektion von Objekten deren Struktur (Attribute), Verhalten (Operationen) und Beziehungen (Assoziationen und Vererbungsstrukturen)*“.

Die Klassensymbole im Klassendiagramm folgen dabei der angegebenen Syntax.

Als Beispiel (Abbildung 17) sei an dieser Stelle die Klasse „Konfiguration Linküberprüfung“ angegeben, deren Aufgabe es ist, die notwendigen Testparameter für die Überprüfung eines Uniform Resource Locators (URL’s) aufzunehmen.

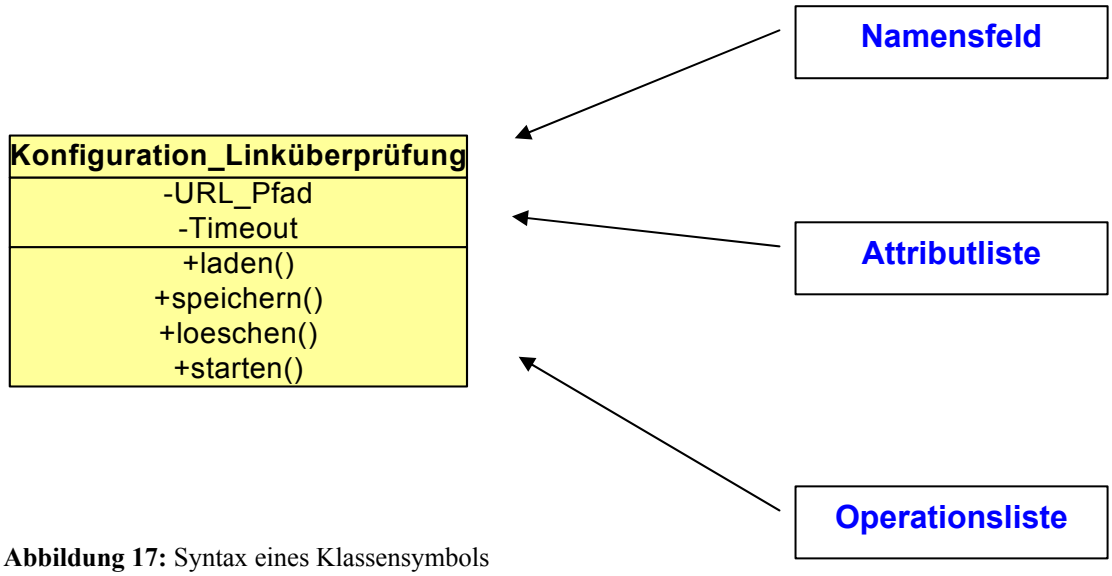


Abbildung 17: Syntax eines Klassensymbols

Weiterhin wurde die Klasse noch um die relevanten Methoden bzw. Operationen ergänzt, welche mit den angegebenen Attributwerten diverse Operationen ausführen.

Neben der Möglichkeit mit Hilfe des Klassendiagramms, die logische Struktur der Anwendungssoftware zu modellieren, ist es mit Hilfe des Klassendiagramms leicht möglich, anhand der Attribute und Operationen, erste Hinweise auf die jeweiligen Interaktionselemente der Benutzungsoberfläche zu erhalten.

[53] Siehe (**Balzert**, 2001**,** S. 5)

Im nachfolgenden Beispiel (Abbildung 18) wird wieder auf die Klasse „Konfiguration Linküberprüfung“ zurückgegriffen.

Beispiel:

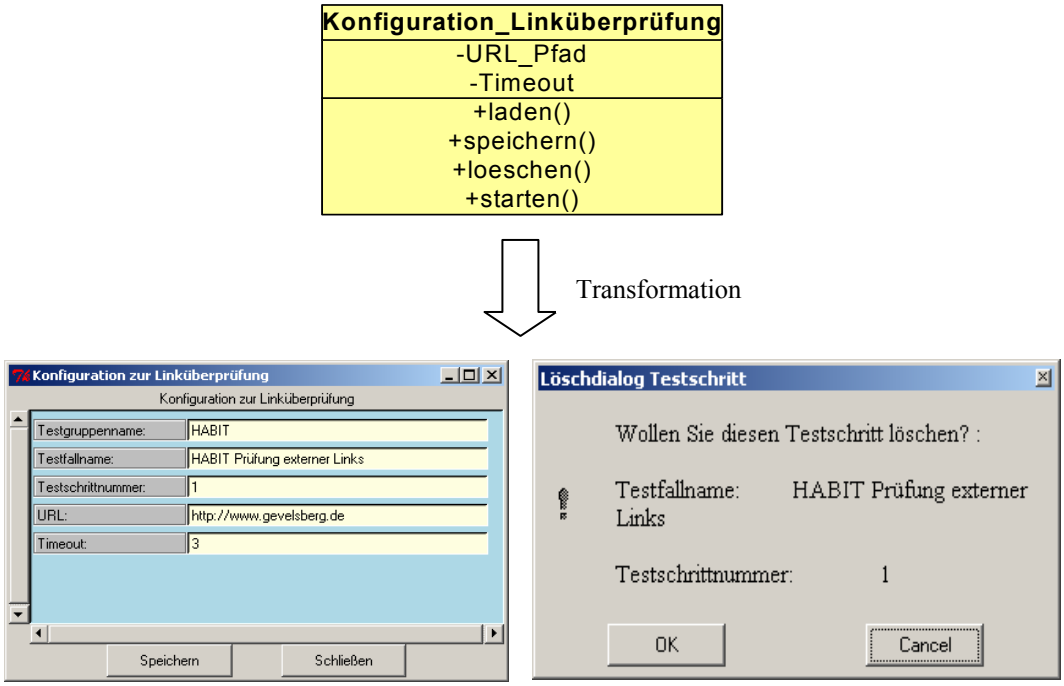


Abbildung 18: Klassentransformation in eine Benutzungsoberfläche

Die Operationen „laden ()“ und „starten ()“ sind jeweils mit Interaktionselementen aus dem Hauptfenster von VirTest verknüpft. Dort wird zum Beispiel die Methode „starten ()“ direkt mit einem Druckschalter aufgerufen. Während die Operation „laden ()“ aus der Klasse „Konfiguration Linküberprüfung“, durch Anklicken eines Listboxeintrages aus der Testschrittliste aktiviert wird.

Die Funktion „speichern()“ und die Attribute „URL“ und „Timeout“ sind direkt über die Eingabefelder beziehungsweise über den entsprechenden Button im Fenster „Konfiguration zur Linküberprüfung“ (siehe oben links) abgebildet worden.

Die Funktion „loeschen()“ hingegen findet sich in einem gesonderten Löschdialog wieder (siehe oben rechts), welcher über eine Art Kontextmenü ebenfalls im Hauptfenster aufgerufen werden kann. Der Button mit der Beschriftung „Schließen“ im Fenster „Konfiguration zur Linküberprüfung“ ist nicht in der entsprechenden Klasse zu finden, da es sich hierbei um eine Funktion handelt, die ausschließlich mit der Bedienung der Benutzeroberfläche in Verbindung steht.

Das folgende Klassendiagramm (Abbildung 19) gibt Ausschnittsweise die logische Struktur der Anwendungssoftware VirTest Version 1.2 wieder, welche für den Use Case „Testdurchführung“ notwendig ist.

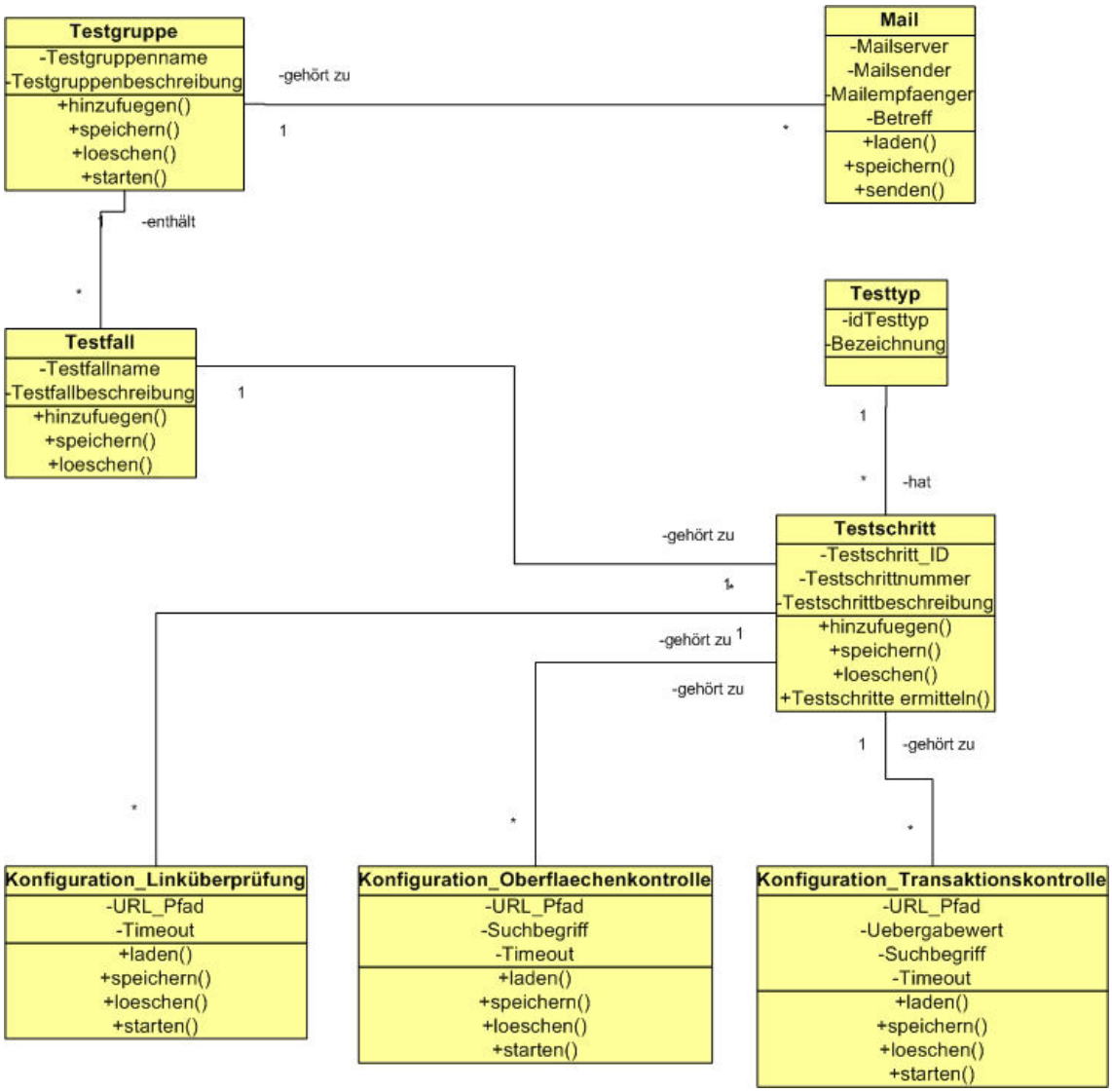


Abbildung 19: Klassendiagramm von VirTest Version 1.2

4.3 Entity-Relationship-Modellierung

Eine weitere Modellierungsmethode, die bei der Erstellung dieser Arbeit Anwendung fand, war die Entity-Relationship-Modellierung. Das Entity-Relationship-Modell (ER-Modell oder ERM) wurde 1976 von P. Chen zur Modellierung von Datenbanken entwickelt.

Im Gegensatz zum vorher genutzten Klassendiagramm wird im ER-Modell kein Verhalten (Operationen und Botschaften) der Software modelliert.

Das Ziel der ER-Modellierung ist in diesem Fall vielmehr die Modellierung der persistent zu speichernden Daten und deren Beziehungen untereinander. Ausgehend von dem zuvor erstellten Klassendiagramm, welches eine gute Übersicht über die Daten und Funktionen der Anwendungssoftware liefert, wurde mit Hilfe des ER-Modells in dieser Arbeit die „Feinspezifikation" der Datenspeicherung vorgenommen.

Eine gute Einordnung des ER-Modells aus der Sicht der Anwendungsentwicklung liefert zunächst das folgende *ANSI-3-Ebenen-Modell*[54]. Es unterscheidet folgende drei Abstraktionsebenen beziehungsweise *Sichten*:[55]

- physische Sicht auf die Daten (interne Ebene)
- logische Gesamtsicht der Daten (konzeptionelle Ebene)
- Benutzersichten auf die Daten (externe Ebene)

Externe Ebene

Die externe Ebene enthält generell die Benutzersicht auf die Daten beziehungsweise den Ausschnitt auf die Gesamtdaten. Bei der Anwendungssoftware VirTest werden in dieser Ebene keine speziellen Benutzersichten eingerichtet, da fast alle Daten von dem jeweiligen Benutzer konfigurierbar und nutzbar sein sollen. Es gibt daher kaum Einschränkungen hinsichtlich der Modifizier- und Sichtbarkeit der Daten.

[54] Bereits 1975 wurde vom ANSI (American National Standards Institut) und SPARC (Standards Planning and Requirements Computing) ein Standard verabschiedet, nach denen sich die Architektur von Datenbanken richtet.
[55] Vgl. (**Faeskorn**, 2001**,** S. 35)

Konzeptionelle Ebene

Diese Ebene beinhaltet die logische Darstellung der Gesamtansicht der Daten in einem speziellen Datenmodell. Bei der Anwendungssoftware VirTest wird durch die Wahl der Datenbank MySQL, auf das in der Praxis am weitesten verbreitete Datenmodell, dem relationalen Datenmodell zurückgegriffen.

Interne Ebene

Die Interne Ebene beinhaltet Informationen über die Art und Weise, wie die Datenstrukturen auf den zumeist physikalischen Datenträger abgelegt sind.

Die nachfolgende Abbildung 20 fasst die zuvor angesprochenen Ebenen einer Datenbankarchitektur aus der Anwendungssicht zusammen:

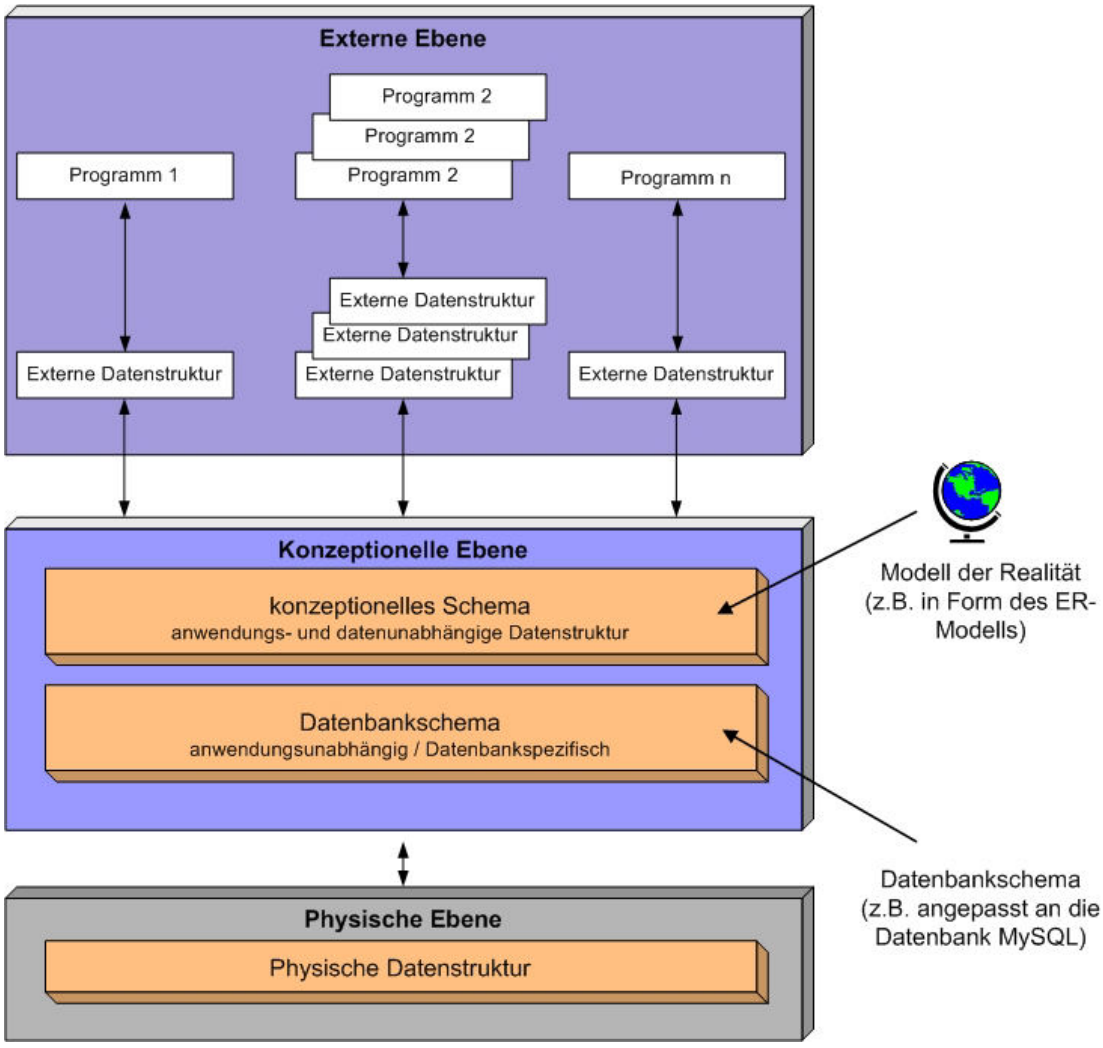


Abbildung 20: Datenbankarchitektur aus Anwendungssicht

4.3.1 ER-Modell von VirTest

Nachfolgend ist das ER-Modell von VirTest Version 1.2 (Abbildung 21), welches mit Hilfe des Case Tools DBDesigner 4 in Krähenfussnotation erstellt wurde, dargestellt.

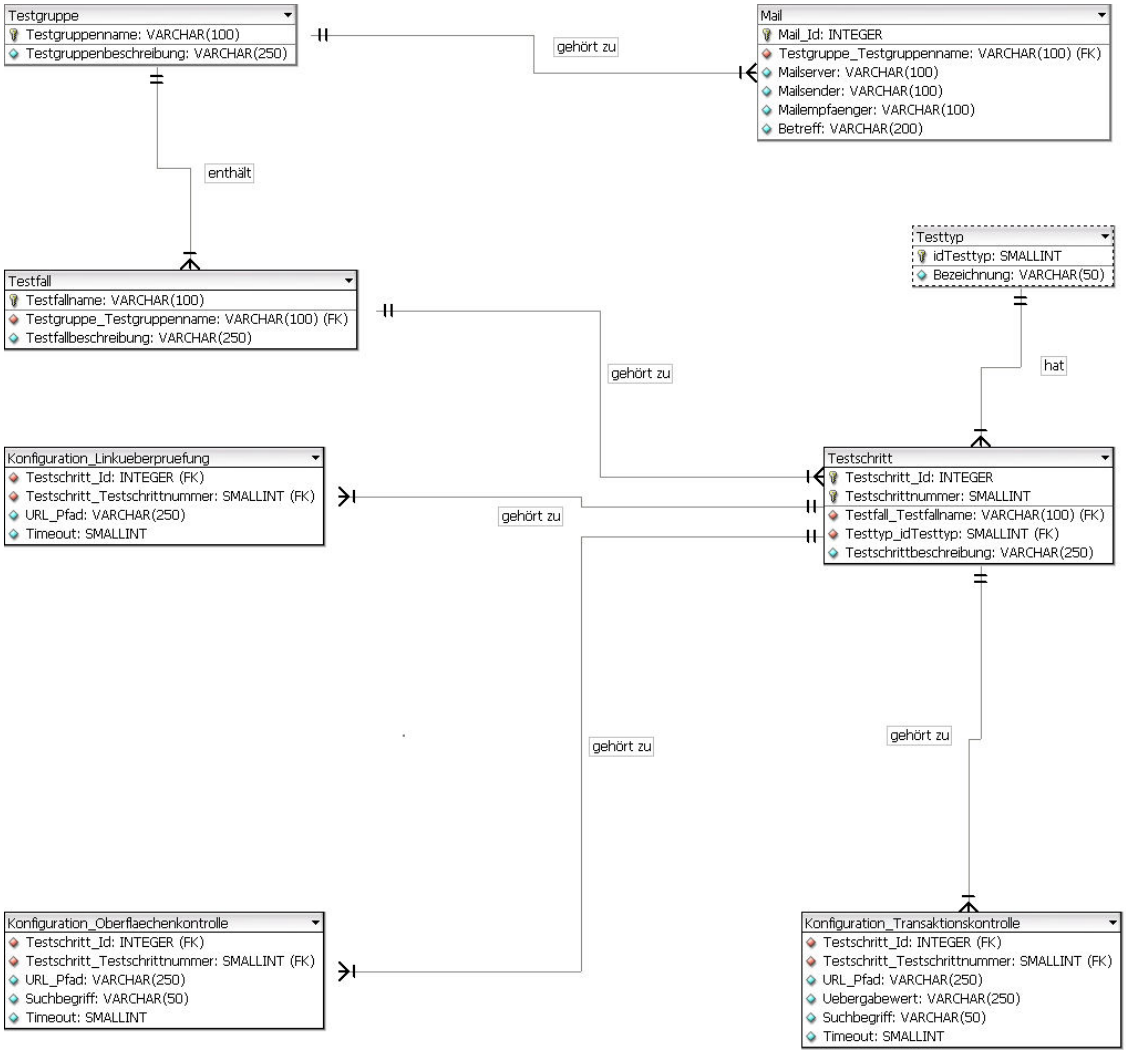


Abbildung 21: ER-Modell von VirTest Version 1.2

4.3.2 Erläuterungen zum VirTest ER-Modell

Bei der Erstellung dieses ER-Modells und dem daraus resultierenden Datenbankschema wurde die 1. Normalform umgesetzt. Das heißt, alle Spalten beinhalten nur atomare Werte und sie können nicht in zwei oder mehr Unterspalten zerlegt werden. Der Ansatz die 2. Normalform anzustreben, wurde verworfen, um die Anzahl der Tabellen möglichst gering zu halten und die Daten, welche offensichtlich als Testparameter für die Anwendung zusammengehören, in einer Tabelle speichern zu können.

Die Gefahr von Änderungsanomalien beim Einfügen beziehungsweise beim Löschen von Daten besteht nicht, da der Benutzer durch die Eingabemasken und die Löschdialoge bestimmten Restriktionen unterworfen ist. Auch der erhöhte Speicherbedarf durch teilweise redundante Daten ist vernachlässigbar gering. Weiterhin verringert sich durch die geringere Anzahl der Tabellen, die Anzahl der notwendigen Join-Verknüpfungen innerhalb des Perl-Skripts, was wiederum zu einer Erhöhung der Abfragegeschwindigkeit führt.

Ein weiterer wesentlicher Vorteil nur die 1. Normalform umzusetzen, zeigte sich während der Implementierung, da es relativ einfach ist sowohl das ER-Modell als auch das Perl-Skript um weitere Testtypen und deren relevanten Daten zu ergänzen. Dies hat speziell für den HABIT den Vorteil, schnell weitere Funktionen dem Prototypen hinzufügen zu können.

Aufgrund der Tatsache, dass die Datenstruktur von VirTest hierarchisch ausgerichtet ist, weist das ER-Modell an vielen Stellen existenzabhängige Entitäten (schwache Entitäten) auf. Dabei handelt es sich um Entitäten, welche in ihrer Existenz von einer anderen, übergeordneten Entität abhängig und oft nur in Kombination mit dem Schlüssel der übergeordneten Entität eindeutig identifizierbar sind.[56]

Als Beispiel sei hier die Entität Testfall aufgeführt, welche ebenfalls nicht mehr existiert, wenn die übergeordnete Entität Testgruppe gelöscht wird. Leider wurde mit Hilfe des CASE-Tools DBDesigner 4.0 keine Notationsmöglichkeit gefunden, um dies im ER-Modell kenntlich zu machen. Der einzige Hinweis ist im Table Editor des CASE-Tools durch eine entsprechende Markierung in einer Checkbox zu finden.

[56] Vgl. (**Kempler**, 2001**,** S. 46)

5 Systemarchitektur VirTest

5.1 Architektur

Bei der Architektur von VirTest handelt es sich um eine 2-Schichten Architektur (two-tier architecture). Das heißt die Architektur besteht aus einer Anwendungsschicht, in der die Benutzungsoberfläche und das Fachkonzept in einer einzigen Schicht miteinander verknüpft sind und aus einer Datenhaltungsschicht.

Die Abbildung 22 gibt die Systemarchitektur von VirTest wieder:

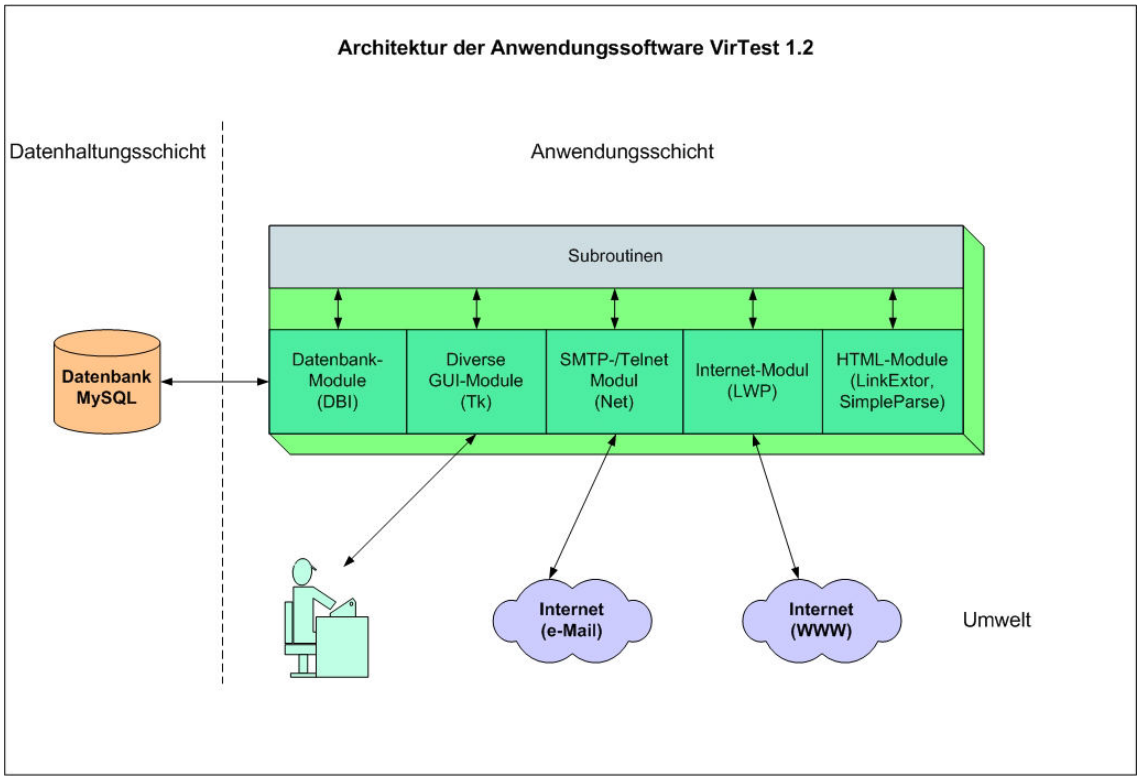


Abbildung 22: Systemarchitektur VirTest

Zur Realisierung der verschiedenen Schnittstellen werden bestimmte Module genutzt, welche die benötigten Methoden bereitstellen. Eine Beschreibung der Funktionsweise der wichtigsten Module, ohne diese eine Umsetzung von VirTest in der vorliegenden Version nicht möglich gewesen wäre, wird in den nachfolgenden Unterkapiteln geliefert. Dazu zählen im Speziellen der Datenbankzugriff, die Entwicklung der grafischen Benutzeroberfläche und der Internetzugriff.

5.1.1 Datenbankzugriff über DBI

Der Zugriff auf eine Datenbank in Perl kann mit Hilfe des so genannten DBI-Moduls (Database Interface) und dem entsprechenden DBD-Modul (Database Driver) realisiert werden.

Dabei definiert das DBI Modul die Programmierschnittstelle, verteilt die Methodenaufrufe an den richtigen Treiber (DBD) und bietet ihm seine Unterstützung an.

Durch die Trennung der *Treiber*[57] vom DBI-Modul wird eine hohe Flexibilität erreicht, so dass derzeit fast jede moderne Datenbank unterstützt wird. Ein weiterer Vorteil ist, dass durch den Austausch des Treibers eine Portabilität der Anwendung auf ein anderes Datenbankprodukt möglich wird.

Die Abbildung 23 verdeutlicht den Datenfluss zwischen einem Perl-Skript und verschiedenen Datenbanken.

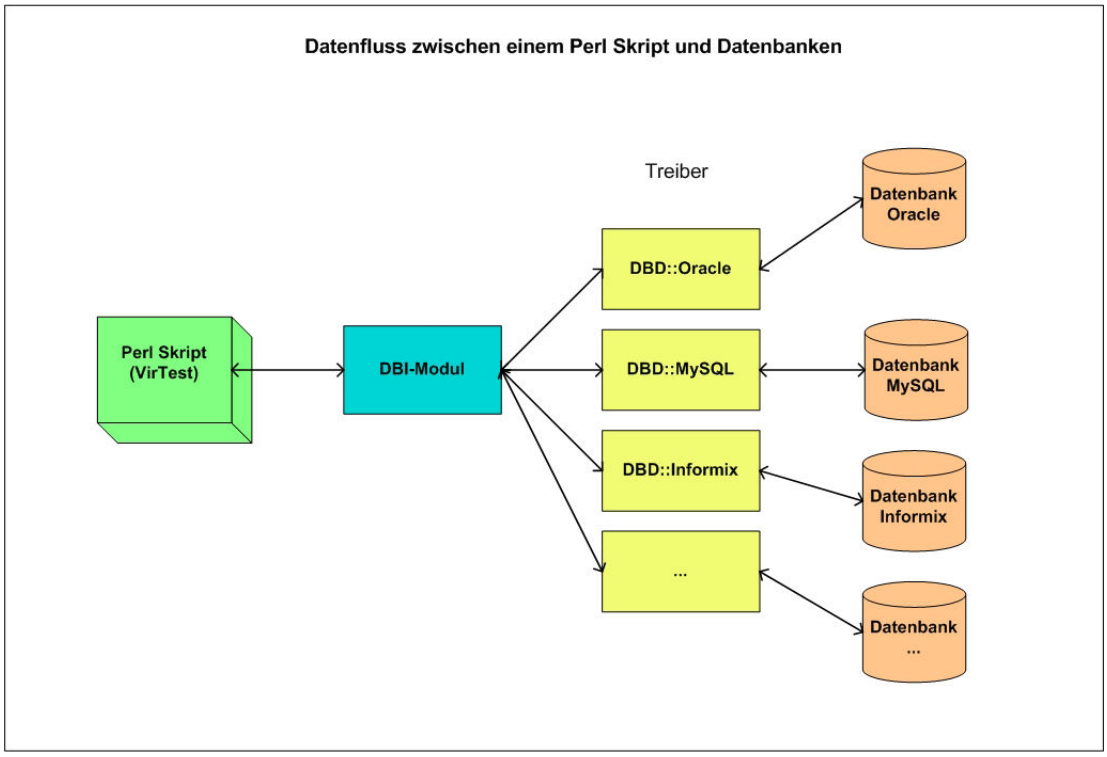


Abbildung 23: Datenfluss zwischen einem Perl Skript und Datenbanken

[57] Es gibt Treiber für nahezu jede moderne Datenbank. Hierzu zählen zum Beispiel Treiber für die Datenbankprodukte der Firmen Oracle, Informix, mSQL, MySQL, Sybase, DB2 und PostgreSQL.

5.1.2 DBI-Datenbank Handle

Das DBI-Modul definiert drei Arten von Objekten, die für die Nutzung von Datenbanken benötigt werden. Diese werden als Handle bezeichnet und repräsentieren die Möglichkeit auf bestimmte Objekte zuzugreifen. Dazu zählen das Treiber-Handle, das Datenbank-Handle und das Statement-Handle.

Diese Handles folgen einer Hierarchie, die nachfolgend dargestellt ist (Abbildung 24).

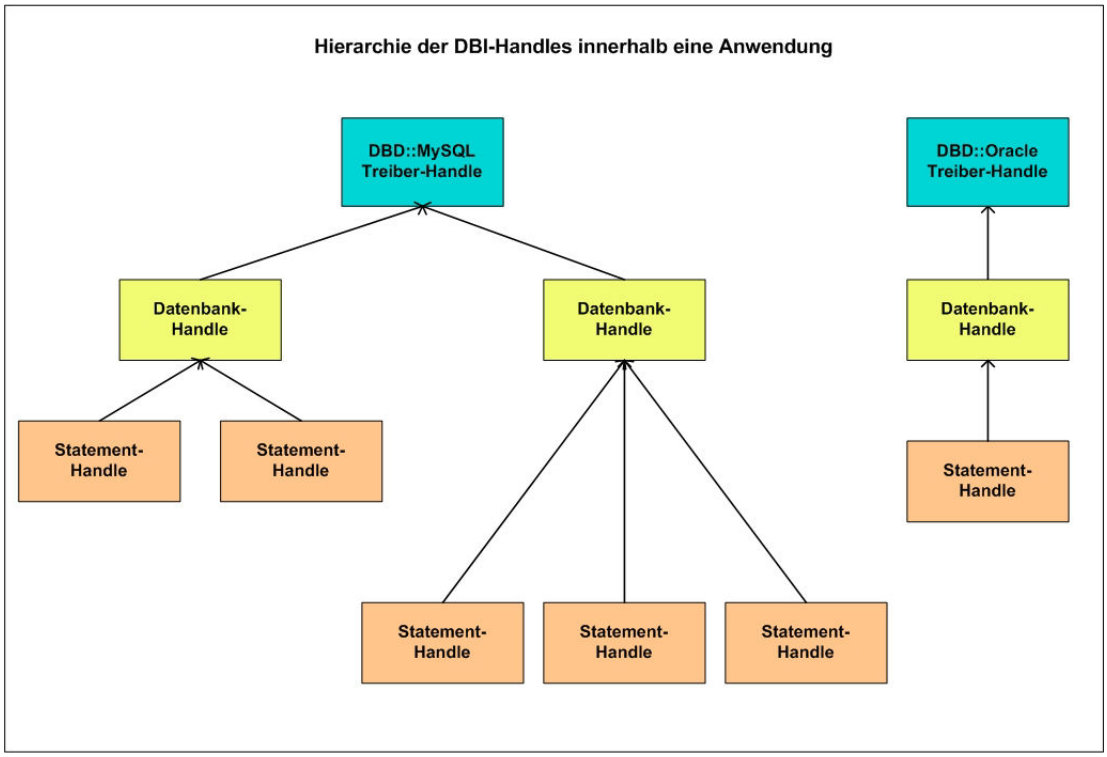


Abbildung 24: DBI-Handles innerhalb eine Anwendung

Mittels dieser Hierarchie können autarke Handles auf verschiedenen Ebenen und zwischen verschiedenen Datenbanksystemen innerhalb einer Anwendung erzeugt werden.

Dadurch wird es möglich sowohl eine große Anzahl verschiedener Datenbanken als auch eine große Anzahl von Datenbankverbindungen mit den entsprechenden SQL-Statements herzustellen. Von dieser Möglichkeit wurde auch bei der Implementierung von VirTest Gebrauch gemacht.

Treiber-Handle:

Unter einem Treiber-Handle wird ein Objekt verstanden, welches einen geladenen Datenbanktreiber repräsentiert und dementsprechend erzeugt wird, wenn der Treiber vom DBI-Modul geladen und initialisiert wird. Dieser Treiber ist anfangs der einzige Kontakt den das DBI-Modul zur Datenbank hat. Es gibt jedoch in dieser Phase noch keine Verbindung zur Datenbank.

Da dieser Treiber durch seinen Handle komplett verkapselt wird, gibt es keinerlei Einschränkungen für das gleichzeitige Laden mehrerer Treiber. Die Treiber-Handles werden in der Regel vom DBI-Modul selbstständig erzeugt und verwendet.[58]

Datenbank-Handle:

Das Datenbank-Handle wird zur Herstellung einer einzelnen gekapselten Verbindung zur Datenbank genutzt. Dazu wird, wie in der nachfolgenden Abbildung 25 dargestellt, die Methode „connect ()“ verwendet.

```
# Erzeugung eines Datenbank-Handles und Zuweisung zu einer Variable
$dbh= DBI-> connect ($data_source,…);
```

Abbildung 25: Erzeugung eines Datenbank-Handles

Die Erzeugung eines solchen Datenbank-Handles ist die Voraussetzung, um im Anschluss SQL-Kommandos an eine Datenbank schicken zu können. Die Anzahl der Datenbank-Handles ist, genauso wie bei den anderen Handle-Typen auch, lediglich durch die Systemressourcen beziehungsweise durch die verwendete Datenbank begrenzt.

Statement-Handle:

Das Statement-Handle dient zur Verkapselung von einzelnen SQL-Kommandos, die von der Datenbank ausgeführt werden sollen. Hier ist es ebenfalls möglich, dass „beliebig“ viele Instanzen gleichzeitig erzeugt werden können, um SQL-Kommandos an eine Datenbank zu schicken.

[58] Vgl. (**Bunce**, 2001**,** S. 83ff)

5.2 Grafische Oberfläche per Perl/Tk

5.2.1 Überblick über Perl/Tk

Um den Anwender von VirTest eine angemessene und zeitgemäße Nutzung der Software zu ermöglichen, wurde das *Tk-Modul*[59] genutzt. Mit Hilfe dieses Moduls ist es möglich, Perl-Skripte mit einer grafischen Benutzerschnittstelle (GUI) zu versehen, wie es mittlerweile bei der Nutzung von Windows-Programmen und anderer Anwendungssoftware üblich ist.

Das heißt es können Fenster mit Interaktionselementen wie Menüs, Druckschaltern, Auswahllisten, Eingabefeldern und anderen grafischen Komponenten ausgestattet werden, die bei der Nutzung des Tk-Moduls generell als *„Widgets“*[60] bezeichnet werden.

Diese mittels Perl/Tk erzeugten Benutzerschnittstellen beruhen auf einer Hierarchie von Interaktionselementen (Widgets), in der an oberster Stelle jeweils das Hauptfenster steht.

Dabei muss jedes Widget ein Vater-Widget haben, dass es bei der Erzeugung und während seiner Lebensdauer überwacht und in der Applikation verwaltet.

Das Hauptfenster wird durch ein Vater-Widget gebildet, welches weitere Widgets enthalten kann. Ein Vater-Objekt kann dabei viele Kinder haben, aber ein Kind nur einen Vater.

[59] Das Tk-Modul wurde ursprünglich von John Ousterhout als Grafik Toolkit für die Programmiersprache TCL entwickelt und später von Nick Ing-Simmons nach Perl portiert.
[60] Die Bezeichnung „Widgets“ (engl.) bedeutet Schaltflächen.

Die Abbildung 26 zeigt die verschiedenen Widget-Grundtypen und deren Anordnung in der Hierarchie.

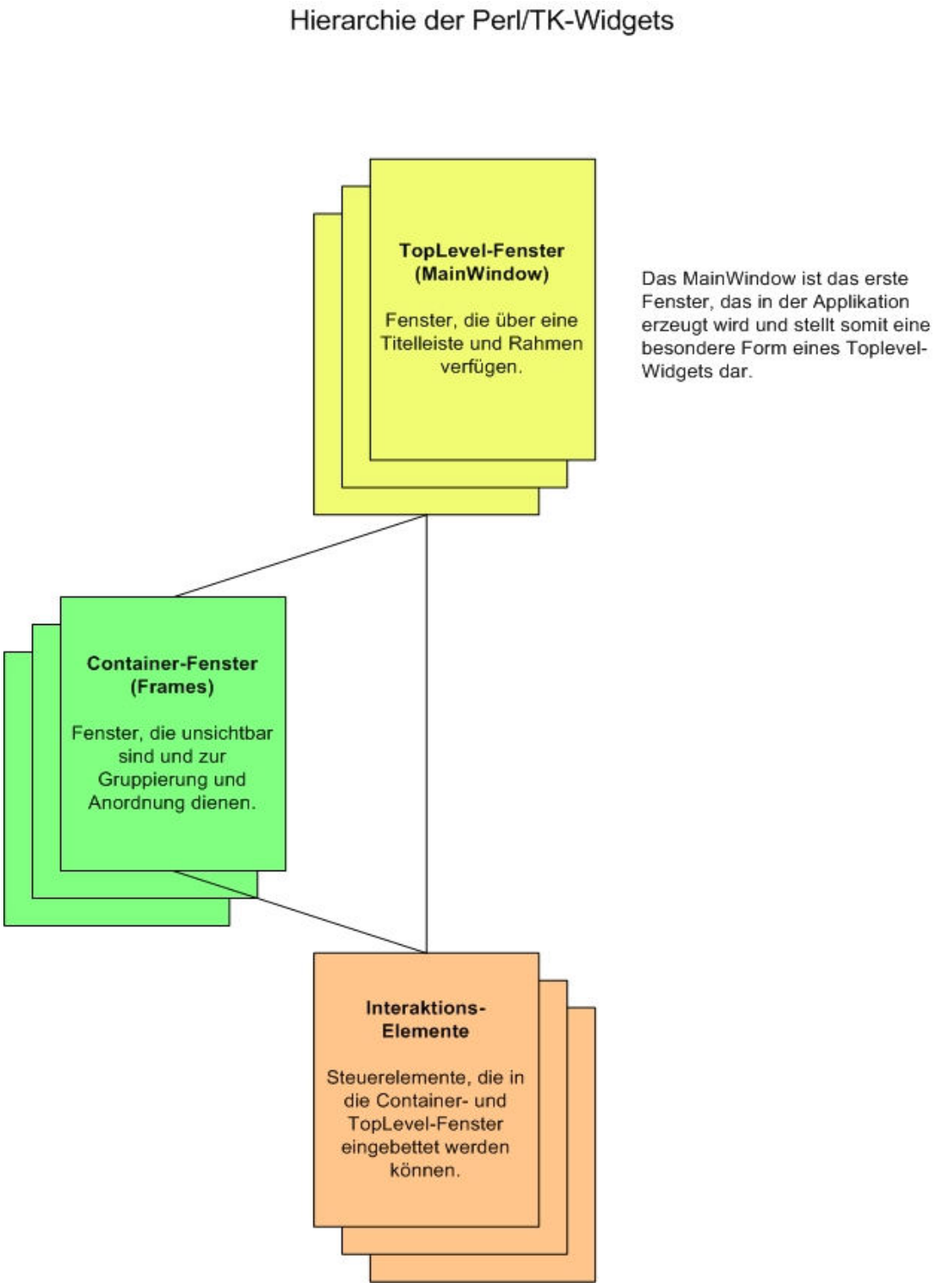


Abbildung 26: Hierarchie der Perl/Tk-Widgets

Um ein solches Hauptfenster (Vater-Objekt) für ein Perl-Skript zu erzeugen, müssen folgende Codezeilen in das Skript eingetragen werden (Abbildung 27):

```
# Erzeugung eines Hauptfensters (Vater-Objekts)

use Tk;                                  # Einbindung des Tk-Moduls
$Mainwindow = MainWindow -> new();       # Erzeugung eines Vater-Objekts
```

Abbildung 27: Erzeugung eines Hauptfensters (Vater-Objekts)

Damit im Programm mit dem erzeugten Hauptfenster gearbeitet werden kann, das heißt zum Beispiel weitere Kinder-Objekte erzeugt werden können, liefert der new-Konstruktor eine Referenz auf das MainWindow-Objekt zurück.

Mit Hilfe dieser Referenz (gespeichert in $MainWindow) ist es jetzt möglich, auf das Hauptfenster (Vater-Objekt) der Anwendung zu zugreifen.

Um den zuvor erzeugten Hauptfenster zum Beispiel einen Druckschalter hinzuzufügen, müssen folgende Code-Zeilen in das Perl-Skript integriert werden (Abbildung 28):

```
# Erzeugung eines Druckschalters im Hauptfenster (Vater-Objekt)

use Tk;                                  # Einbindung des Tk-Moduls
$Mainwindow = MainWindow -> new();       # Erzeugung eines Vater-Objekts
$schalter=$Mainwindow-> Button (         -text     =>“Bezeichnungstext“,
                                         -command        => sub{exit0});
$schalter -> pack();                     # Anzeige des Druckschalters
MainLoop;
```

Abbildung 28: Erzeugung eines Druckschalter im Hauptfenster

In diesem Fall wird der Druckschalter bereits bei der Erzeugung mit bestimmten Eigenschaften, wie dem Bezeichnungstext und einem Unterprogramm ausgestattet. Die Methode „pack“ spezifiziert einen so genannten *Geometrie-Manager*[61], welcher für die Anordnung und Größe des Widgets in seinem Vater-Objekt verantwortlich ist.

Der Aufruf der Methode MainLoop startet eine Ereignisschleife, auf die im nächsten Kapitel näher eingegangen wird.

[61] Neben dem Geometrie-Manager „pack“, gibt es noch die Geometrie-Manager „grid“ und „place“, welche je nach Anforderung der Applikation Anwendung finden.

5.2.2 Ereignisverarbeitung in Perl/TK

Damit die erzeugten Widgets mit einer entsprechenden Funktion auf Eingaben reagieren können, werden diese mit so genannten Ereignissen (Events) verknüpft. Ereignisse sind in diesem Zusammenhang normalerweise verschiedene Kombinationen aus Maus- und/oder Tastaturinteraktionen, die entweder seriell oder parallel vom Anwender durchgeführt werden.

Diese Ereignisse werden in Perl/TK-Programmen in einer Ereignisschleife abgearbeitet, welche durch die Methode „MainLoop“ gestartet wird. Innerhalb der Ereignisschleife wird ermittelt, ob und welche Art von Eingabe gemacht wurde und anschließend bestimmt, welche Unterprogramme *(Callback-Funktionen)*[62] aufgerufen werden sollen.

Jedes Widget in Perl/Tk hat seine eigenen Default-Verknüpfungen (Bindungen), das heißt zum Beispiel, wenn ein Druckschalter angeklickt wird, wird eine festgelegte Callback-Funktion aufgerufen. Dabei handelt es sich um Default-Verknüpfungen, die mit dem Erzeugen des Widgets angelegt werden.

Um den Aufruf von Callback-Funktionen durch sein Widget noch flexibler zu gestalten, bietet Perl/Tk noch die Möglichkeit Widgets mit selbst definierten Ereignissen zu verknüpfen.

Von dieser Möglichkeit wird auch in dieser Arbeit Gebrauch gemacht, zum Beispiel bei der Anzeige von Hilfetexten in einem Textfeld, abhängig von der Cursorposition oder beim Aufruf von quasi kontextsensitiven Eingabemasken, beim Betätigen der rechten Maustaste über den Listboxen. Möglich wird diese Funktionalität über die Nutzung der der so genannten „bind“-Methode.

[62] Das diese speziellen Unterprogramme Callback-Funktionen heißen, liegt an der Tatsache, dass nicht der Anwender selbst diese Unterprogramme aufruft, sondern die interne Tk-Modul Nachrichtenverarbeitung.

Als Beispiel (Abbildung 29) für eine selbst definierte Bindung sei an dieser Stelle eine Listbox angeführt, welche beim Betätigen der rechten Maustaste (sofern sich der Cursor über dieser befindet) die Callback-Funktion „Testergaenzung_Testschritt“ über eine Referenz aufruft.

```
# Erzeugung einer selbst definierten Verknüpfung (Bindung)
$lb_Testschritt-> bind('<Button-3>',\&Testergaenzung_Testschritt);
```

Abbildung 29: Erzeugung einer Bindung in Perl/Tk

5.3 Internetzugriff über LWP

Das wohl wichtigste Modul zur Realisierung der Anwendung VirTest ist das Modul mit dem Namen *LWP*[63]. Die Abkürzung LWP steht für "Library for World Wide Web in Perl" und zeigt bereits anhand des Namens die Ausrichtung der integrierten Methoden auf.

Genau gesagt, ist LWP eigentlich kein einzelnes Modul, sondern eher eine Sammlung von gleich gearteten Modulen und kann daher als Bibliothek aufgefasst werden.

Mit Hilfe dieser Bibliothek ist es möglich eine Fülle von Aufgaben, die dass World Wide Web betreffen, zu automatisieren.

Einen Ausschnitt aus dieser Funktionsvielfalt liefert die nachfolgende Auflistung:

- Erzeugung von Web-Clients, die Webseiten von anderen Servern holen
- Daten an Web-Server schicken
- Dokumente von anderen Servern spiegeln

Innerhalb dieser Modulsammlung ist das objektorientierte Packet „UserAgent“, das für diese Anwendung wichtigste Modul. Mit Hilfe dieses Moduls ist es möglich, ein so genanntes *User-Agent-Objekt*[64] zu erzeugen.

Mit diesem User-Agent-Objekt können zum Beispiel die HTTP-Anfragen „GET“, „HEAD“ und „POST“ durchgeführt werden, um Informationen über das spezifizierte Web-Objekt zu holen beziehungsweise Information an dieses zu schicken.

Im nachfolgenden Unterkapitel wird exemplarisch auf die Nutzung des „UserAgent“-Moduls aus der LWP-Bibliothek eingegangen.

[63] Das Modul wurde von Gisle Aas und Martijn Koster entwickelt.
[64] Allgemein ist ein User-Agent eine Schnittstelle zwischen dem Benutzer und der entsprechenden Netzwerkanwendung. Im Falle der WWW-Anwendung ist zum Beispiel der jeweilige Browser ein User-Agent.

5.3.1 Nutzung des User-Agent Moduls

Für Web-Client-Anwendungen, wie in diesem Fall bei VirTest, wird grundsätzlich zuerst ein User-Agent-Objekt angelegt. Anschließend wird ein HTTP-Request-Objekt erzeugt und konfiguriert, dass an die request-Methode des User-Agent-Objekts übergeben wird.

Der angefragte Server gibt dann zum entsprechenden User-Agent-Objekt die Antwort in Form eines Response-Objekts zurück.

Die nachfolgende Abbildung 30 erklärt diesen Mechanismus:

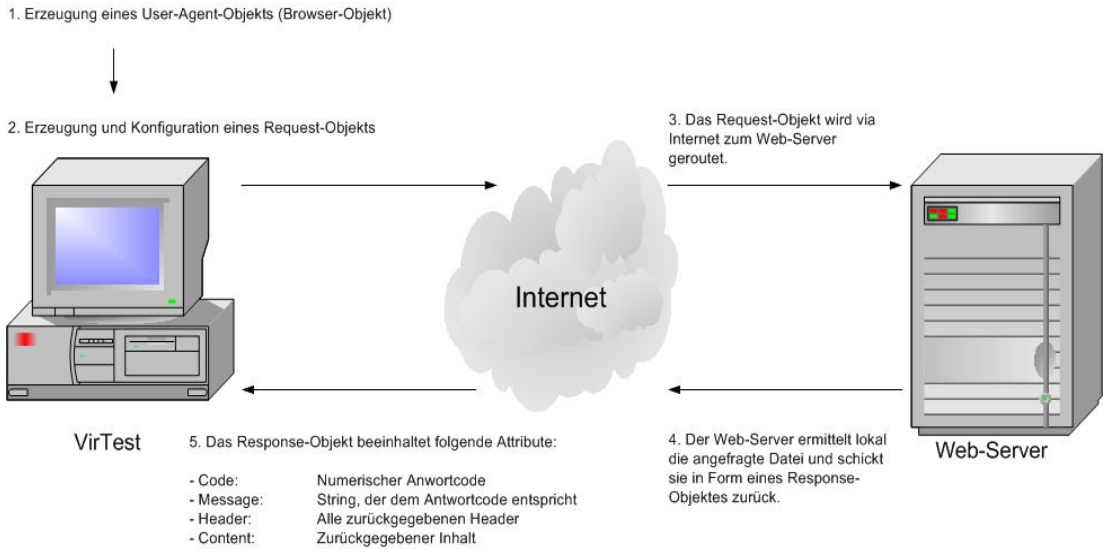


Abbildung 30: Allgemeine Nutzung der LWP-Bibliothek

Die notwendigen Implementierungsschritte zur Nutzung der LWP-Bibliothek werden nachfolgend exemplarisch anhand einer HTTP-Transaktion aufgeführt (Abbildung 31).

In diesem Fall werden Informationen über die Startseite der Stadt Hagen per URL ’http://www.hagen.de’ mit Hilfe der HTTP-Transaktion (GET) in einem Response Objekt gespeichert.

```
# Nutzung der LWP-Bibliothek
use LWP::UserAgent;                    # Einbindung des User Agent LWP-Moduls
$url=’http://www.hagen.de’;            # Definition eines URL’s
# 1. Erzeugung eines User-Agent-Objekts
$User_Agent_Objekt = LWP::UserAgent->new( );

# 2. Erzeugung und Konfiguration eines Request-Objekts
$Request_Objekt = HTTP::Request->new(’GET’,$url);

# 3. Übergabe an das User-Agent-Objekt und Erzeugung eines Response-Objekts
$Response_Objekt= $User_Agent_ObjeKt-> request($Request_Objekt);
```

Abbildung 31: Implementierung einer URL-Abfrage mittels LWP

Über den Inhalt der HTML-Seite hinaus werden Informationen über:

- den numerischen Antwortcode,
- den String, der dem Antwortcode entspricht
- und alle zurückgegebenen Kopfzeilen (Header)

im Response-Objekt gespeichert.

Mit Hilfe dieser Informationen ist es möglich eine weitere Analyse der entsprechenden HTML-Seite, wie zum Beispiel ein Parsing nach bestimmten Suchbegriffen, durchzuführen.

6 Implementierung von VirTest

Im folgenden Kapitel wird näher auf die Implementierung von VirTest eingegangen. Dazu werden zunächst die verwendeten Werkzeuge und CASE-Tools besprochen und deren wichtigsten Merkmale aufgezeigt. Anschließend wird mit Hilfe von Quellcodeauszügen exemplarisch die Implementierung der Aspekte Datenhaltung und -zugriff, grafische Oberfläche, sowie Systemfunktionalitäten erklärt. Sofern Probleme bei der Implementierung aufgetreten sind, werden diese ebenfalls in den jeweiligen Unterkapiteln angesprochen.

6.1 Verwendete Werkzeuge und CASE-Tools

6.1.1 Komodo 3.0

Die Implementierung des Perl-Quellcodes wurde vollständig mit der integrierten Entwicklungsumgebung Komodo 3.0 der Firma ActiveState unter dem Betriebssystem Windows 2000 Professional umgesetzt. Hierbei handelt es sich um eine *IDE*[65], die speziell für eine Reihe von Skriptsprachen wie Perl, Python, TCL und PHP ausgelegt ist.

Die folgende Abbildung 32 zeigt das Hauptfenster der IDE:

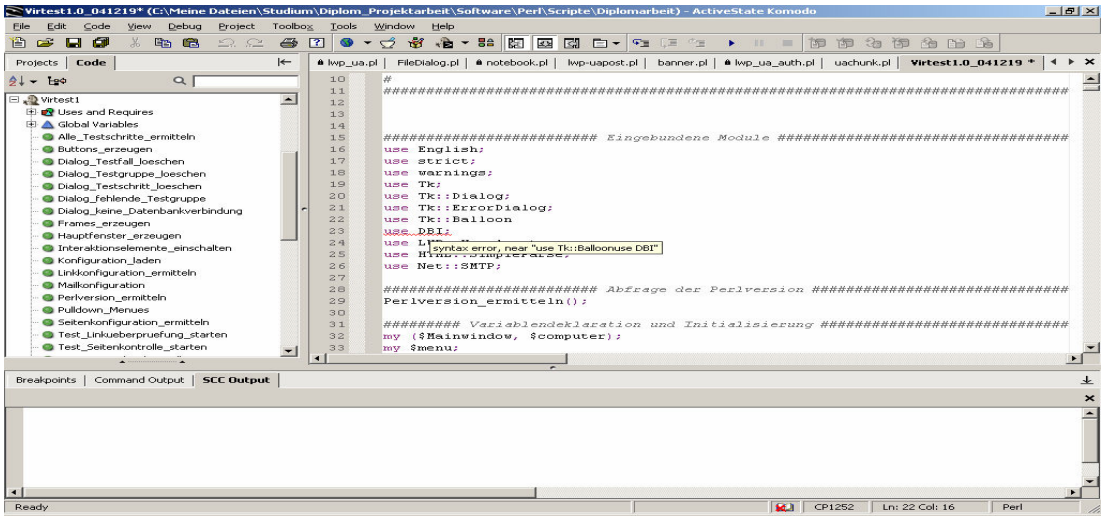


Abbildung 32: Hauptfenster der Komodo 3.0 IDE

[65] IDE ist die Abkürzung für Integrated Development Environment

Komodo bietet im Vergleich zu einem einfachen Editor in Verbindung mit einem Interpreter eine Reihe von Vorteilen, von denen sich besonders die folgenden Merkmale als hilfreich erwiesen:

- Syntax-Highlightning, die farbliche Differenzierung von Schlüsselwörtern
- Syntaxprüfung, während der Eingabe des Quelltexts und Anzeige des Warnungs- beziehungsweise Fehlertextes
- Bei der Bildung von Codeblöcken wird die entsprechende schließende Klammer farblich markiert
- Anzeige von versehentlich doppelt deklarierten Variablen innerhalb eines Geltungsbereiches
- Auskommentieren selektierter Bereiche per Menüeintrag
- Mittels eine Code-Browsers ist es leicht möglich, sich zu den gewünschten Unterroutinen zu bewegen

Der einzige Nachteil von Komodo, welcher durch entsprechende Vorsichtsmaßnahmen zu keinen weiteren Nachteilen führte, war das sporadische Auftreten von Stabilitätsproblemen.

Diese konnten nur durch ein Beenden der IDE mit Hilfe des Windows-Taskmanagers „behoben“ werden. Die genaue Ursache der Stabilitätsprobleme konnte vom Autor nicht ermittelt werden.

Zusammenfassend lässt sich sagen, dass die Implementierung von VirTest durch die Verwendung von Komodo Version 3.0 wesentlich vereinfacht wurde.

6.1.2 DBDesigner 4

Die Erstellung des ER-Modells, also des konzeptionellen Schemas und des Datenbankschemas, erfolgte mit dem CASE-Tool DB Designer 4. Hierbei handelt es sich um ein visuelles CASE-Tool, welches speziell auf die Zielplattform MySQL ausgerichtet ist und daher bei der Erzeugung des entsprechenden Datenbankschemas keine weiteren Anpassungen notwendig waren. Leider war hier ebenfalls die Nutzung des Tools teilweise durch Stabilitätsprobleme beeinträchtigt. Im Großen und Ganzen überwogen auch hier die Vorteile, welche nachfolgend aufgeführt sind:

- Kostenlos und frei verfügbar unter der GNU GPL Lizenz
- Einfache Erstellung und Modifikation des ER-Modells per GUI im Drag- & Drop-Verfahren
- Datenbanksynchronisation, die Integration des erstellten ER-Modells in die MySQL-Datenbank
- Schnelle Modifikation von Beispieldaten im Abfragemodus mit Hilfe der Benutzeroberfläche
- Erzeugung und Speicherung von SQL-Anweisungen im Abfragemodus
- Export des ER-Modells als Grafik

Nachfolgend ist das Hauptfenster im Entwurfsmodus dargestellt (Abbildung 33):

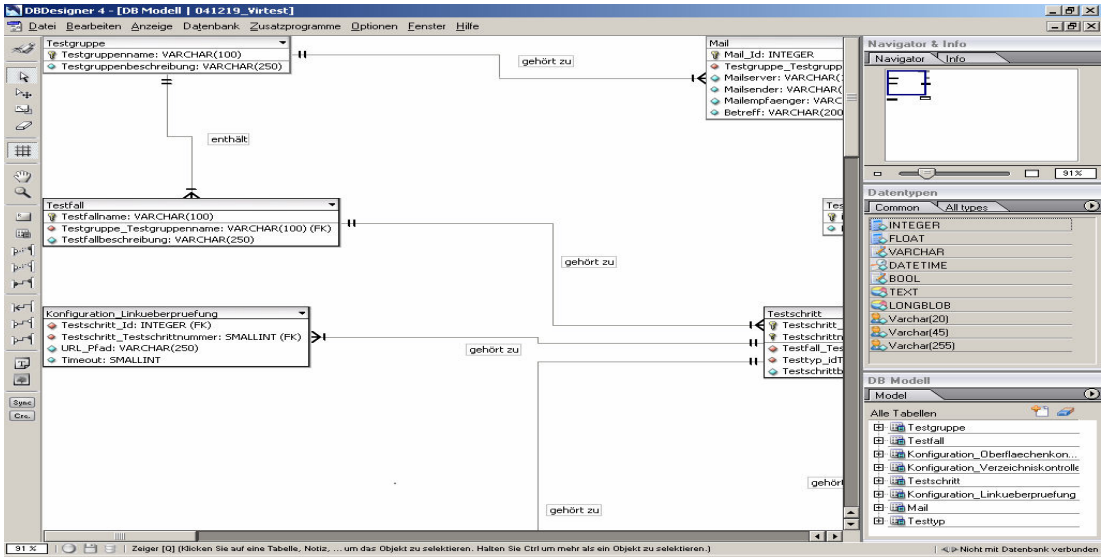


Abbildung 33: Hauptfenster des CASE-Tools DBDesigner 4.0

6.2 Datenhaltung und Schnittstellenrealisierung

Die Basis zur Datenhaltung aller relevanten Daten für die Anwendungssoftware VirTest bildet das RDBMS MySQL. In diesem Datenbanksystem werden alle Daten gespeichert, welche für die Testdurchführung benötigt werden. In den nachfolgenden Unterkapiteln, werden exemplarisch Quellcode-Auszüge dargestellt, die:

- Die Erzeugung der Datenbank und
- Die Schnittstelle zu Perl verdeutlichen.

6.2.1 Erzeugung des Datenbankschemas

Zur Erzeugung des Datenbankschemas wurde das CASE-Tool DBDesigner genutzt, welches aus dem erstellten ER-Modell die entsprechenden SQL-Anweisungen generiert. Exemplarisch sei an dieser Stelle die Erzeugung der Tabelle „Testschritte“ dargestellt, welche eine zentrale Rolle im Datenmodell erfüllt.

Diese Tabelle „kennt“ alle sich in VirTest befindlichen Testschritte und deren Zugehörigkeit zum jeweiligen Testfall und Testtyp.

Um diese Tabelle in das Zieldatenbanksystem MySQL zu integrieren, muss folgende SQL-Anweisung durchgeführt werden (Abbildung 34).

```
-- Erzeugung der Tabelle Testschritt

CREATE TABLE Testschritt (
  Testschritt_Id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  Testschrittnummer SMALLINT UNSIGNED NOT NULL,
  Testfall_Testfallname VARCHAR(100) NOT NULL,
  Testtyp_idTesttyp SMALLINT UNSIGNED NOT NULL,
  Testschrittbeschreibung VARCHAR(250) NULL,
  PRIMARY KEY(Testschritt_Id, Testschrittnummer),
  INDEX Testschritt_FKIndex1(Testfall_Testfallname),
  INDEX Testschritt_FKIndex2(Testtyp_idTesttyp)
);
```

Abbildung 34: Erzeugung der Tabelle Testschritt

6.2.2 Perl Datenbank-Schnittstelle

Um auf die MySQL-Datenbank zugreifen zu können wird, das Database Interface-Modul (DBI) und der entsprechende MySQL-Datenbanktreiber (DBD) im Perl-Skript genutzt.

In den nachfolgenden Quellcode-Auszügen wird nun gezeigt, wie die Datenbankschnittstelle realisiert ist und wie mit verschiedenen Methoden der Zugriff und die Manipulation von Daten vorgenommen wurden.

Von der Grundstruktur her erfolgt dieser Ablauf grundsätzlich sequentiell, angefangen bei der Herstellung der Datenverbindung, bis hin zur Beendigung. Daher wird dieser Ablauf auch nachfolgend als Sequenz dargestellt.

1. Einbindung des DBI-Moduls in den Perl-Quelltext

Die Einbindung des objektorientierten DBI-Moduls erfolgt, wie die Einbindung der anderen Module, mit Hilfe des use-Befehls (siehe Kapitel 3, S. 43).

2. Herstellung der Datenbankverbindung

Nachdem das DBI-Modul geladen wurde, muss zunächst eine Verbindung mit der MySQL Datenbank hergestellt werden. Dazu wird die Methode „connect ( )“ genutzt, die ein Datenbank-Handle Objekt zurückgibt, das in der Variable $dbh gespeichert wird (Abbildung 35).

Dabei müssen der „connect ( )“-Methode bestimmte Parameter (siehe Kommentarzeilen) übergeben werden, um die Verbindung korrekt herstellen zu können.

```
…
$dbh=DBI->connect
(
"DBI:mysql:host=localhost;          # Angaben über die Datenbank
database=virtest",                  # Name der Datenbank
"root",                             # User-Name
"",                                 # Passwort
{PrintError=>0,                     # Unterdrückung der Warnung
RaiseError=>1}                      # Abbruch bei Fehler
);
```

Abbildung 35: Herstellung der Datenbankverbindung

Mittels dieses Datenbank-Handles werden nun alle weiteren Aktionen, die diese Datenbankverbindung betreffen, durchgeführt.

Es bestand ebenso die Möglichkeit weitere Datenbank-Handles zur gleichen MySQL-Datenbank aufzubauen. Dies hat sich während der Implementierung als nicht notwendig herausgestellt.

Mit Hilfe des erzeugten Datenbank-Handles können jetzt „beliebig“ viele Statement-Handles erzeugt werden.

Als Beispiel wird hier das Statement-Handle dargestellt, welches zur Initialisierung des Listenfeldes im Hauptfenster mit den bereits in der Datenbank befindlichen Testgruppennamen genutzt wird.

Dazu wird mit der „prepare ( )“-Methode dem DBI-Modul mitgeteilt, welche SQL-Anweisung es zu erwarten hat. Daraufhin wird die SQL-Anweisung von der Datenbank überprüft und der entsprechende Ablaufplan erzeugt. Anschließend wird mit der „execute ( )“-Methode die Datenbank angewiesen den Befehl durchzuführen.

Sofern die Anweisung ohne Probleme durchgeführt werden konnte, setzt die Datenbank einen Zeiger auf die erste Zeile der Ergebnismenge. Die Adresse dieses Zeigers wird im Statement-Handle „$sth_testgruppen_initialisierung“ gespeichert (Abbildung 36).

3. Erzeugung eines Statement-Handles

```
…

$sth_testgruppen_initialisierung=$dbh -> prepare
(
        "SELECT Testgruppenname FROM Testgruppe
        ORDER BY Testgruppenname"
);

$sth_testgruppen_initialisierung->execute();

…
```

Abbildung 36: Erzeugung eines Statement-Handles

Im nächsten Schritt werden die Daten mit der Methode *„fetchrow_array ( )“*[66] eingelesen. Da diese Methode immer nur eine Zeile der Ergebnismenge als Array liefert, werden alle Daten mit Hilfe einer while-Schleife ermittelt und dem Array „@zeile_testgruppen“ zugewiesen.

Das erzeugte Array wird schließlich der Perl/Tk-Methode „insert ( )“ zur Übernahme und zur Darstellung in das Listenfeld übergeben (Abbildung 37).

4. Nutzung des Statement-Handles mittels der Methode „fetchrow_array()“

```
…

while   ( my @zeile_testgruppen=$sth_testgruppen_initialisierung
              ->fetchrow_array())
        {
                    $lb_Testgruppenliste->insert
                    ('end',@zeile_testgruppen);
        }
```

Abbildung 37: Nutzung des Statement-Handles

Wenn das Statement-Handle nicht mehr benötigt wird, sollte es mit der nachfolgenden „finish ()“-Methode beendet werden (Abbildung 38).

Falls dies die einzige SQL-Anweisung im Perl-Skript wäre, so sollte abschließend das Datenbank-Handle mit der „disconnect ( )“-Methode beendet werden.

5. Beendigung des Statement –und des Datenbank-Handles

```
…

$sth_testgruppen_initialisierung->finish();
…

$dbh->disconnect();
…
```

Abbildung 38: Beendigung des Datenbank-Handles

[66] Die Methode „fetchrow_array ( )“ ist nur eine von vielen Methoden, die dass DBI-Modul zur Verfügung stellt. Es kann hier nicht auf die Funktionsweise aller Methoden eingegangen werden.

6.3 Grafische Oberfläche

Zur Implementierung der grafischen Oberfläche wurde das Tk-Modul genutzt. Dieses Modul stellte alle Methoden bereit, die zur Erzeugung der Benutzerschnittstelle notwendig waren.

Die Applikation VirTest besteht aus einem Hauptfenster, das mit einer Menüleiste versehen und zur vorrangigen Mausbedienung ausgelegt ist. Im oberen Bereich des Hauptfensters befinden sich die drei Listboxen, mit denen die Testkonfigurationen aufgebaut und modifiziert werden können.

Dazu wird einfach die rechte Maustaste über der entsprechenden Listbox betätigt, wodurch sich die entsprechenden Dialogfenster öffnen. Hier kann der Benutzer die Daten in die dafür vorgesehenen Eingabefelder eintragen.

Um in diesen Dialogfenstern eine schnellere Eingabe zu gewährleisten, ist hier ebenfalls eine Navigation mittels der Tabulator-Taste möglich.

Mit Hilfe der bind ()-Methode war es weiterhin möglich, einige dieser Interaktionselemente um weitere Funktionen zu erweitern, um das kontextsensitive Verhalten zu erzeugen. So ist zum Beispiel zum Löschen eines gewählten Eintrages aus einem Listenfeld, die Betätigung der <Strg>-Taste in Verbindung mit der linken Maustaste möglich.

Im unteren Bereich des Hauptfensters befindet sich der Ausgabebereich. Hier werden die Ergebnisse der Testdurchführung und die relevanten Testeingangsparameter dargestellt.

Auch hier befindet sich ein kontextsensitives Menü, welches folgende Funktionen ermöglicht:

- Editorfunktionen des Ausgabetextes
- Suchfunktionen im Ausgabetext
- Funktionen für die Zwischenablage

Durch die Verwendung der Zwischenablage wird in diesem ersten Prototypen, die Möglichkeit geboten, die Testergebnisse zu speichern oder anderen Personen per e-Mail zur Verfügung zu stellen (Wunschkriterium aus dem Pflichtenheft).

Zur Umsetzung der Benutzerschnittstelle wurden hauptsächlich folgende Interaktionselemente genutzt, um eine möglichst intuitive Bedienung der Software zu ermöglichen.

- Hauptfenster mit Pulldown-Menüs
- Rahmen
- Rollbalken
- Druckschalter
- Eingabefeld
- Optionsfeld
- Textfeld
- Dialogfenster
- Listenfeld

Das erzeugte Hauptfenster hat folgendes Aussehen (Abbildung 39):

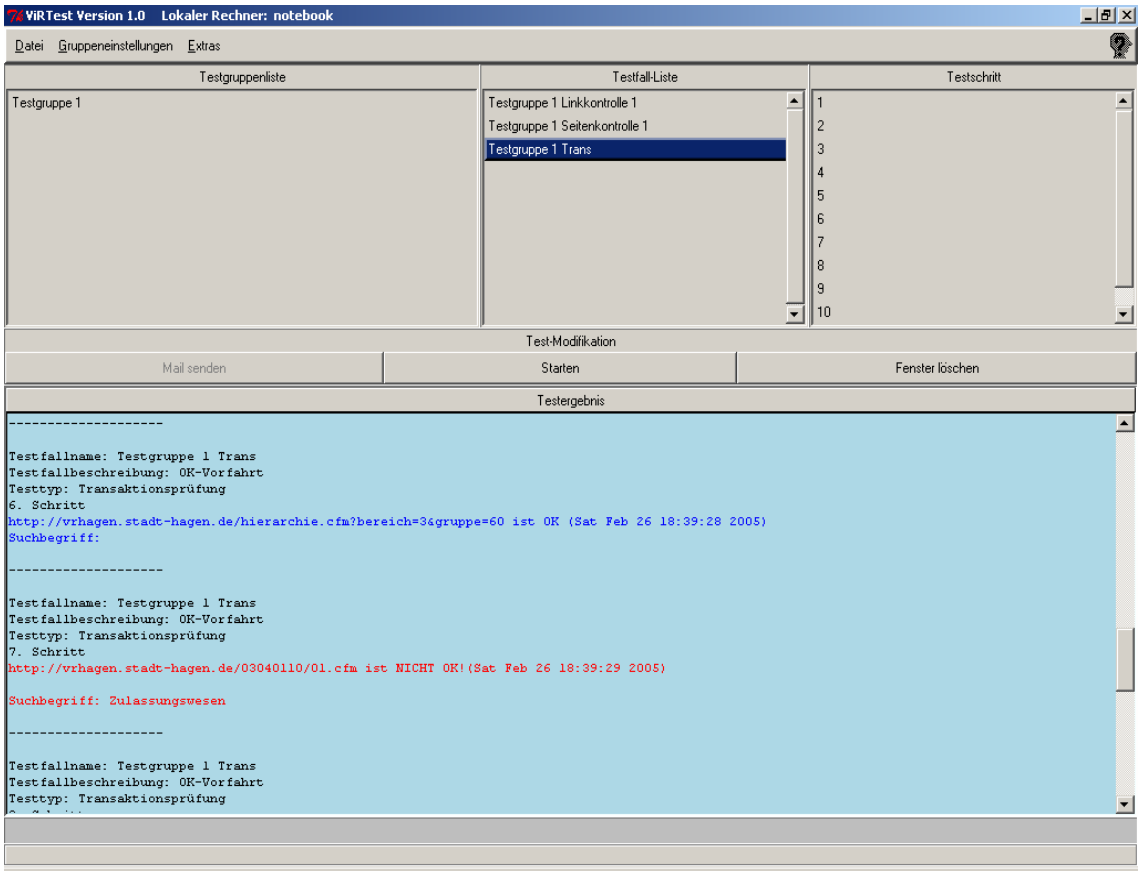


Abbildung 39: Hauptfenster von VirTest Version 1.2

Um Probleme bei der Datenhaltung zu vermeiden, wurden die Fensterdialoge die eine Änderung der Testparameter zur Folge haben können, grundsätzlich *anwendungsmodal*[67] ausgelegt und wichtige Zusatzinformationen bei der Darstellung dynamisch eingebunden.

Zur Erzeugung des vollständigen Hauptfensters ist der Aufruf folgender Unterfunktionen notwendig (Abbildung 40):

```
# Hauptprogramm
…
Hauptfenster_erzeugen();
…

# Erzeugung der Listen
Testschrittliste_darstellen();
Testfalliste_darstellen();
Testgruppenliste_darstellen();
…
# Ende des Hauptprogramms


sub Hauptfenster_erzeugen{
        $computer =`hostname`;
        chomp($computer);
        $Mainwindow = MainWindow->new();
        $Mainwindow->title("VirTest Version 1.2     Lokaler Rechner:  $computer");
        $Mainwindow->geometry("900x600+0+0");
        # Erzeugung eines Balloons
        $balloon = $Mainwindow->Balloon();
        # Pulldown-Menues
        Pulldown_Menues();
        # Hilfelabel erzeugen und Frames packen
        Frames_erzeugen();
        Buttons_erzeugen();
}
```

Abbildung 40: Erzeugung des Hauptfensters von VirTest

[67] Anwendungsmodal bedeutet, dass es dem Benutzer von VirTest zwar möglich ist, zu einer anderen Anwendung zu wechseln, jedoch innerhalb von VirTest der Dialog zunächst beendet werden muss.

6.4 Funktionen

VirTest ermöglicht in der ersten Version datenbankgestützt folgende Funktionen:

- Kontrolle von Weboberflächen
- Kontrolle von HTML-Links
- Kontrolle des einwandfreien Transaktionsvermögens

Zusätzlich wurde eine Funktion integriert, die ausgehend von einem Basis-URL alle vorhandenen Links kontrolliert. Diese Funktion wurde allerdings nicht datenbankgestützt umgesetzt, da diese für ad-hoc Kontrollen vorgesehen ist. Exemplarisch wird nun auf die Implementierung der HTML-Linküberprüfung eingegangen.

6.4.1 Internetzugriff

Die Testdurchführung „Kontrolle von Links“ wird mittels verschiedener Module und Unterfunktionen realisiert. Sofern die Testparameter in der MySQL-Datenbank administriert sind, werden die nachfolgenden Funktionen zur Umsetzung der Linküberprüfung genutzt. Nachdem der Button „Starten“ in der GUI betätigt wurde, wird zunächst geprüft, ob eine Testgruppe ausgewählt wurde. Wenn dies nicht der Fall ist, wird ein Dialogfenster aufgerufen, welches auf diesen Umstand hinweist.

Sobald eine gültige Testgruppe gewählt wurde, wird zur Funktion „Testdurchfuehrung_starten ()“ gesprungen und der Funktion der Name der Testgruppe mitgeteilt (Abbildung 41).

```
# Erzeugung des Buttons „Starten“ und Aufruf der Testdurchführung
…
$button_ElementTesten=$ModifikationsFrame->Button( -text =>"Starten",
                                                   -activeforeground=>'blue',
                                                   -command => sub{
if($ausgewaehlte_Testgruppe eq ""){ \&Dialog_fehlende_Testgruppe($Mainwindow)}
else{
        $button_ElementTesten->configure(-state=>"disabled");
        $button_ElementTesten->Busy(-recurse=>1);
        $id=$Mainwindow->after(2000,\&Testdurchfuehrung_starten
        (\$ausgewaehlte_Testgruppe));
}
}       )->pack (-side=>'left', -fill=> 'x',-expand=>1);
```

Abbildung 41: Erzeugung des Druckschalters „Starten“

Die Funktion „Testdurchfuehrung_starten ()“ ermittelt alle relevanten Testschritte der übergebenen Testgruppe mit Hilfe der Funktion „Alle_Testschritte_ermitteln()“ und ordnet diese den zugehörigen Use-Cases anhand der eindeutigen Testtyp-Id (hier 1) zu.

Für die entsprechenden Use-Cases werden wiederum die relevanten Testparameter aus der Datenbank herausgesucht (hier die Funktion „Linkkonfiguration_ermitteln ()“) und der jeweiligen Funktion (hier „Test_Linkueberpruefung_starten()“) übergeben (Abbildung 42).

```
# Ermittlung und Zuweisung aller zu einer Testgruppe gehörenden Testschritte

sub Testdurchfuehrung_starten{
my $ausgewaehlte_TestgruppeRef=shift;
my ($Alle_TestschritteRef, $LinkkonfigurationRef,
$SeitenkonfigurationRef, $TransaktionskonfigurationRef);
my $i=0;

$Alle_TestschritteRef=Alle_Testschritte_ermitteln($$ausgewaehlte_TestgruppeRef);
if (($#{$Alle_TestschritteRef}) >= 0){
$ErgebnisText->insert('end',"Testgruppenname: $Alle_TestschritteRef->[$i][0]\n");
$ErgebnisText->insert('end',"Testgruppenbeschreibung: $Alle_TestschritteRef->[$i][5]\n\n");

for ($i=0; $i <= $#{$Alle_TestschritteRef}; $i++){
$ErgebnisText->insert('end',"Testfallname: $Alle_TestschritteRef->[$i][1]\n");
$ErgebnisText->insert('end',"Testfallbeschreibung: $Alle_TestschritteRef->[$i][6]\n");

if ($Alle_TestschritteRef->[$i][3] ==1){ ## Testtyp-Id für die Linkueberpruefung
$ErgebnisText->insert('end',"Testtyp: Linküberprüfung\n");
$ErgebnisText->insert('end',"Testschrittbeschreibung:
$Alle_TestschritteRef->[$i][7]\n");
$ErgebnisText->insert('end',"$Alle_TestschritteRef->[$i][4]. Schritt\n");
$LinkkonfigurationRef=Linkkonfiguration_ermitteln($Alle_TestschritteRef->[$i][2]);
Test_Linkueberpruefung_starten
($LinkkonfigurationRef->{URL_Pfad}, $LinkkonfigurationRef->{Timeout});
                    }
….
```

Abbildung 42: Aufruf der Funktion „Testdurchfuehrung_starten()“

Innerhalb der Funktion „Test_Linkueberpruefung_starten ()“ wird jetzt das User-Agent Objekt erzeugt, konfiguriert und zur Testdurchführung herangezogen (Abbildung 43).

```
# Durchführung der Linküberprüfung mit Hilfe des User-Agent-Objekts
…

sub Test_Linkueberpruefung_starten{

my $uebergebene_URL_Pfad=shift;
my $uebergebene_Timeout=shift;
my ($ua,$anfrage, $antwort, $antwort_code,$antwort_message,$antwort_content);

$ua = LWP::UserAgent->new();
$ua->timeout($uebergebene_Timeout);

$anfrage = HTTP::Request->new('GET', $uebergebene_URL_Pfad);

$antwort = $ua->request($anfrage);

$antwort_code=$antwort->code();
$antwort_message=$antwort->message();
$antwort_content=$antwort->content();


if ( $antwort->is_error()) {
if($uebergebene_URL_Pfad){
        $ErgebnisText->insert('end',$uebergebene_URL_Pfad .= " ist NICHT OK
        (" . localtime() .")\n",'FoundError');

        }
        $ErgebnisText->insert('end',"Error-Code: $antwort_code\n",'FoundError');
        $ErgebnisText->insert('end',"Error-Message: $antwort_message\n",'FoundError');
        $ErgebnisText->insert('end',"-" x 20);
        $ErgebnisText->insert('end',"\n" x 3);

}
else {
        $uebergebene_URL_Pfad .= " ist OK (" . localtime() .")\n\n";
        $ErgebnisText->insert('end',$uebergebene_URL_Pfad,'URL');
        $ErgebnisText->insert('end',"-" x 20);
        $ErgebnisText->insert('end',"\n\n");
        }
}
}
}
```

Abbildung 43: Aufruf der Funktion „Test_Linkueberpruefung_starten()“

Die Testparameter und -ergebnisse werden sowohl aus der Funktion „Testdurchfuehrung_starten ()“ als auch aus der jeweiligen Testdurchführungsfunktion mittels der insert ()-Methode an das Ausgabefenster ($ErgebnisText) übergeben.

7 Abschluss

7.1 Zusammenfassung

Das Ziel der Diplomarbeit war die Entwicklung einer Anwendungssoftware, welche das so genannte Blackbox-Testverfahren unterstützt. Mit Hilfe der Software sollte es möglich sein, dass Internetportal der Stadt Hagen hinsichtlich bestimmter Grundfunktionen automatisiert zu überprüfen.

Dieses Ziel ist in den wesentlichen Punkten erfüllt worden, wobei allerdings die Erkenntnis, dass Software eigentlich nie fertig sein kann und immer weitere gute Ideen während der Implementierung aufkommen, eine wichtige Erfahrung war.

Um den derzeitigen Stand der Software zu erreichen wurde bei der Implementierung auf die Skriptsprache Perl/Tk und die Datenbank MySQL zurückgegriffen. Der Ansatz die Datenhaltung der Testparameter zunächst mittels komplexer Datenstrukturen, wie zum Beispiel der Verschachtelung von *Hashes*[68] zu realisieren, wurde aufgrund der sehr hohen Komplexität und der daraus resultierenden schwierigen Umsetzbarkeit in einem frühen Projektstadium verworfen. An dieser Stelle noch einmal vielen Dank an Prof. Dr. Faeskorn-Woyke für den wichtigen Tipp.

Dadurch die Wahl der Datenbankspeicherung wurde es möglich auf die ER-Modellierung und das CASE-Tool DBDesigner 4 zurückzugreifen, mit dessen Hilfe die Datenhaltung in der MySQL-Datenbank schnell realisiert werden konnte.

Die Wahl der Skriptsprache Perl/Tk zur Implementierung hat sich nicht in allen Bereichen, als die beste Möglichkeit herausgestellt. Beim Versuch der Modellierung mittels der Unified Modeling Language war es nicht möglich, alle UML-Modellierungsnotationen zu nutzen, da Perl nicht alle objektorientierten Methoden unterstützt.

[68] Ein Hash oder auch assoziatives Array ist ein Datentyp, der eine effiziente Schlüssel-Wertepaarverwaltung ermöglicht. Es handelt sich dabei um ein spezielles Arrays, deren Elemente nicht über einen Index, sondern über eine beliebige Zeichenkette adressiert werden.

Daher konnte die UML nicht durchgängig genutzt werden und es kamen wenige Notationsmöglichkeiten zum Einsatz. Speziell der Einsatz des Klassendiagramms erwies sich als schwierig und es konnten nur Hinweise auf die Erstellung der Benutzungsoberfläche vom Klassendiagramm abgeleitet werden.

Als problematisch erwies sich weiterhin die Tatsache, dass Perl kein *Threading*[69] unterstützt und daher bei der Erstellung des Userinterfaces Probleme auftraten. Da jedoch die Software in der Definitionsphase von vorne herein für eine Einzelplatz-Installation ausgelegt war, ist dieser Umstand akzeptabel.

Als effizient erwies sich hingegen die Nutzung der wichtigsten Module DBI und LWP. Mittels dieser Module konnte ein problemloser Datenbank- und Internetzugriff umgesetzt werden.

Die Annahme, dass eine weite Verbreitung eines bestimmten Interfaces, also die zwischen dem Perl DBI-Modul und der MySQL-Datenbank für eine gewisse Qualität und Stabilität sorgt, hat sich bestätigt.

Auch die in der einschlägigen Perl-Literatur häufig zu findende Nutzung des LWP-Moduls, hat sich bei der Implementierung des Internetzugriffs als sehr hilfreich erwiesen.

Der nun entwickelte Prototyp ist derzeit beim Hagener Betrieb für Informationstechnologie installiert und befindet sich dort im Testbetrieb.

[69] Unter Threading versteht man die parallele Ausführung mehrere Programmfäden (threads). Es gibt zwar für Perl ein Thread-Modul, dieses befindet sich jedoch in einem experimentellen Status, so dass aus Stabilitätsgründen auf die Verwendung verzichtet wurde.

7.2 Fazit und Ausblick

Die Wahl dieses Diplomarbeitsthemas war eine hervorragende Möglichkeit sich mit einem aktuellen und breiten Spektrum von Themen auseinander zusetzen.

Es hat sich ebenfalls gezeigt, dass jeder Bereich für sich, angefangen von der Definitionsphase bis hin zur Implementierung, teilweise eine Spezialisierung in dem jeweiligen Bereich notwendig macht, um tiefer gehende Problemstellungen bewältigen zu können.

Aufgrund des funktionalen und modularen Aufbaus der Anwendungssoftware VirTest ist es relativ einfach das Programm um weitere Funktionen zu ergänzen und den Bedürfnissen des Kunden (HABIT) anzupassen.

Mögliche Ansatzpunkte für eine Erweiterung wären zum Beispiel:

- Ergänzung der Software um einen Web-Recorder, also einer Funktion zur Aufzeichnung der Nutzungsschritte innerhalb eines Browsers
- Integration eines Browsers zur Darstellung der Web-Seiten
- Ergänzung der E-Mail-Funktionalität
- Speicherung der Testergebnisse in einer Datenbank zur statistischen Auswertung und zur Ermittlung der Verfügbarkeit des Virtuellen Rathauses
- Entwicklung einer aktiven Alarmierung aufgrund der Testergebnisse zur zeitnahen Problembehebung

Falls der HABIT sich dazu entscheiden sollte alle genannten Ansatzpunkte zu integrieren, müsste aufgrund der rapide ansteigenden Komplexität der Software noch mal die Wahl der Implementierungssprache überdacht werden, um alle Anforderungen erfüllen zu können.

Glossar

CORBA

Die Common Object Request Broker Architecture kurz CORBA ist eine objektorientierte Middleware, die plattformübergreifende Protokolle und Dienste definiert und von der Object Management Group (OMG) entwickelt wird.

CORBA ermöglicht das Erstellen verteilter Anwendungen in heterogenen Umgebungen. CORBA ist nicht an eine bestimmte Programmiersprache gebunden.

Mittels einer Interface Definition Language (IDL) erstellt man eine formale Spezifikation der Klassen und Objekte sowie sämtlicher Parameter und Datentypen. Diese Schnittstellenbeschreibung wird dann in ein Objektmodell der verwendeten Programmiersprache umgesetzt.[70]

CFML®

CFML ist die Abkürzung für Cold Fusion Markup Language, einer Programmiersprache, welche speziell für den ColdFusion Applikation Server entwickelt wurde und in die Kategorie der Skriptsprachen einzuordnen ist.

Es handelt sich hierbei um eine serverseitige Sprache, die aufgrund der Verwendung von Tags eine große Ähnlichkeit mit der Hypertext Markup Language (HTML) hat.

HTTPS

Das HTTPS-Protokoll ist die sichere Variante von HTTP, die im WWW eine verschlüsselte Datenübertragung zwischen Browser und Server ermöglicht. HTTPS nutzt den SSL-Standard (Secure Socket Layer), einen von Netscape entwickelten, offenen Standard zur gesicherten Übertragung. Ein entsprechender Server würde beispielsweise adressiert mit https://www.nummersicher.de.

Es gibt verschiedene Sicherheitsstufen, je nach Verschlüsselungsstärke. Während es in den USA ein Exportverbot für starke Verschlüsselungstechnologien gibt, wird in Europa die 40-Bit Schlüssellänge zunehmend von der 128-Bit-Verschlüsselung verdrängt.

Software, die nicht aus den USA exportiert wurde, kann auch mehr als 128-Bit unterstützen. Online-Banking und elektronischer Zahlungsverkehr wird in der Regel über HTTPS/SSL gesichert. Es sind neue Protokolle in Entwicklung, unter anderem eine Variante die auf PGP basiert.[71]

[70] Siehe URL: (http://www.net-lexikon.de/CORBA.html [Stand: 15.06.2004])
[71] Siehe URL: (http://www.net-lexikon.de/HTTPS.html [Stand: 15.06.2004])

OSCI

Steht für Online Services Computer Interface. Hierbei handelt es sich um einen Standard, der mit dem Protokoll OSCI-Transport die sichere Übertragung von elektronisch signierten Nachrichten ermöglicht.

XML

Abkürzung für eXtensible Markup Language. XML ist eine Metasprache für die Definition von Dokumenttypen, die ein anpassungsfähiges Datenformat für den Austausch strukturierter Dokumente im Web ermöglicht.

XML ist eine vereinfachte Variante der Sprache SGML - Standard Generalized Markup Language -, die bereits 1986 von der International Standards Organisation (ISO) als Standardmetasprache für die Auszeichnung von Dokumenten installiert wurde.

Während HTML als ein spezieller Dokumenttyp aus SGML entwickelt wurde - eine einfache Markup-Sprache mit vorgegebenen Tags -, behält XML den Charakter einer Metasprache bei und gibt dem Entwickler die Möglichkeit, eigene angepasste Markup-Sprachen - quasi Dialekte - mit eigenen Tags für viele verschiedene Dokumenttypen zu definieren. Die Version XML 1.0 ist seit Februar 1998 vom W3C standardisiert.

Literaturverzeichnis

Alex et al. 2001
ALEX, W.; BERNÖR, G.; ALEX, B: Einführung in Unix. Karlsruhe: Skriptum der Universität Karlsruhe, 2001

Alex 2004
ALEX, Wulf: Einführung in das Internet. Karlsruhe: Skriptum der Universität Karlsruhe, 2004

Balzert 1998
BALZERT, Helmut: Lehrbuch der Software-Technik – Software-Management, Software-Qualitätssicherung. Heidelberg, Berlin: Spektrum, Akad. Verlag, 1998

Balzert 1999
BALZERT, Heide: Lehrbuch der Objektmodellierung - Analyse und Entwurf. Heidelberg, Berlin: Spektrum, Akad. Verlag, 1999

Balzert 2001
BALZERT, Heide: UML kompakt. Heidelberg, Berlin: Spektrum, Akad. Verlag, 2001

Balzert 2000
BALZERT, Helmut: Lehrbuch der Software-Technik – Software-Entwicklung. Heidelberg, Berlin: Spektrum, Akad. Verlag, 2000

Bunce 2001
BUNCE, Tim: Programmierung mit Perl DBI. Köln: O'Reilly Verlag, 2001

Burke 2002
BURKE, Jean: Perl & LWP. Sebastopol: O'Reilly Verlag, 2002 (Online-Edition)

Faeskorn 2001
FAESKORN-WOYKE, Heide: Datenbanken und Informationssysteme – Einführung in die Grundbegriffe der Datenbanken. Land Nordrhein-Westfalen: Lerneinheit des Instituts für Verbundstudien der Fachhochschulen Nordrhein-Westfalens, 2001

Forta 2003
FORTA, Ben; WEISS, Nate: ColdFusion MX Web Application Construction Kit. Berkley: Macromedia Press 2003

Hajji 2000
HAJJI, Farid: Perl Einführung, Anwendungen, Referenz. München: Addison Wesley Longman Verlag, 2000

Herde 2001
HERDE, Georg: Softwaretechnik-Folge 3 – Methoden und Konzepte der Softwareentwicklung. Deggendorf: Skriptum der FH-Deggendorf, 2001

Hinz 2002
HINZ, Stefan: MySQL-Referenzhandbuch (Übersetzung des englischen Originals). Berlin: iConnect, 2002

Kahle et al. 2004
KAHLE, I.; SCHÄFER, D; TIMM, U.: Informationstechnologie in Haushalten- Ergebnisse einer Pilotstudie für das Jahr 2003. Wiesbaden: Statistisches Bundesamt-Pressestelle, 2004

Kempler 2001
KEMPLER, Alfons: Datenbanksysteme- Eine Einführung. München; Wien: Oldenbourg Verlag, 2001

Klinger 2003
KLINGER, Peter: Datenschutzgerechtes eGovernment. Hagen: Informationsschrift der HABIT GmbH, 2003

Lucke 2000
REINERMANN, Heinrich; VON LUCKE, Jörn: Speyerer Definition von Electronic Government. Speyer: Online-Publikation des Forschungsinstituts für öffentliche Verwaltung, 2000

Münz 2002
MÜNZ, Stefan: HTML - Referenz. München: Franzis' Verlage, 2002

Oechsle 2002
OECHSLE, Rainer: TCP/IP Transport und Vermittlung im Internet. Fachhochschule Trier: Fernstudium Informatik, 2002

Schmitz et al. 2003
SCHMITZ, C.; GOLDFUSS, S; CIELEN, P.: Coldfusion MX - Professionelle Anwendungsentwicklung fürs Web. München: Addison-Wesley Verlag, 2003

Sparling 2002
SPARLING, DOUG; WILES, Frank: Perl Module - Effiziente Anwendungsentwicklung mit Modulen. München: Addison-Wesley Verlag, 2002

Walsh 2000
WALSH, Nancy: Einführung in Perl/TK. Köln: O'Reilly Verlag, 2000

Winter 2004
WINTER, MARIO: Vorlesungsfolien aus dem Verbundstudiengang Wirtschaftsinformatik – Fortgeschrittene Softwareentwicklung. Köln: Vorlesungsfolien des SS2004, 2004

Anhang A Pflichtenheft VirTest Version 1.2

A1 Zielbestimmung

Der Hagener Betrieb für Informationstechnologie (HABIT) soll durch die Software in die Lage versetzt werden einen Teil der bisher manuell durchgeführten Funktionstests zur Überprüfung des Internetangebots „Virtuelles Rathaus“ automatisiert durchzuführen.

A1.1 Musskriterien

- Speicherung der Testkonfigurationen
- Modifikationsmöglichkeit der Testkonfigurationen
- Automatische Testdurchführung
- Ausgabe der Testergebnisse und relevanter Testparameter

A1.2 Wunschkriterien

- Grafische Benutzungsoberfläche
- Unterstützung bei der Erstellung der Testkonfigurationen
- Benachrichtigung per E-Mail über die Testergebnisse

A1.3 Abgrenzungskriterien

- Es ist kein Hilfesystem vorgesehen
- Es kein Muti-User-Betrieb und kein Rollenkonzept vorgesehen
- Es ist kein Automatismus für den Start der Testdurchführung vorgesehen, dass heißt die Testdurchführung wird manuell von einem Endbenutzer gestartet

A2 Produkteinsatz

Die Software dient den Mitarbeitern des Hagener Betriebs für Informationstechnologie (HABIT) zur Automatisierung von Funktionstest des Internetangebots.

A2.1 Anwendungsbereiche

Die Software ist zur Unterstützung des Produktionsbetriebs des Hagener Betriebs für Informationstechnologie vorgesehen.

A2.2 Zielgruppen

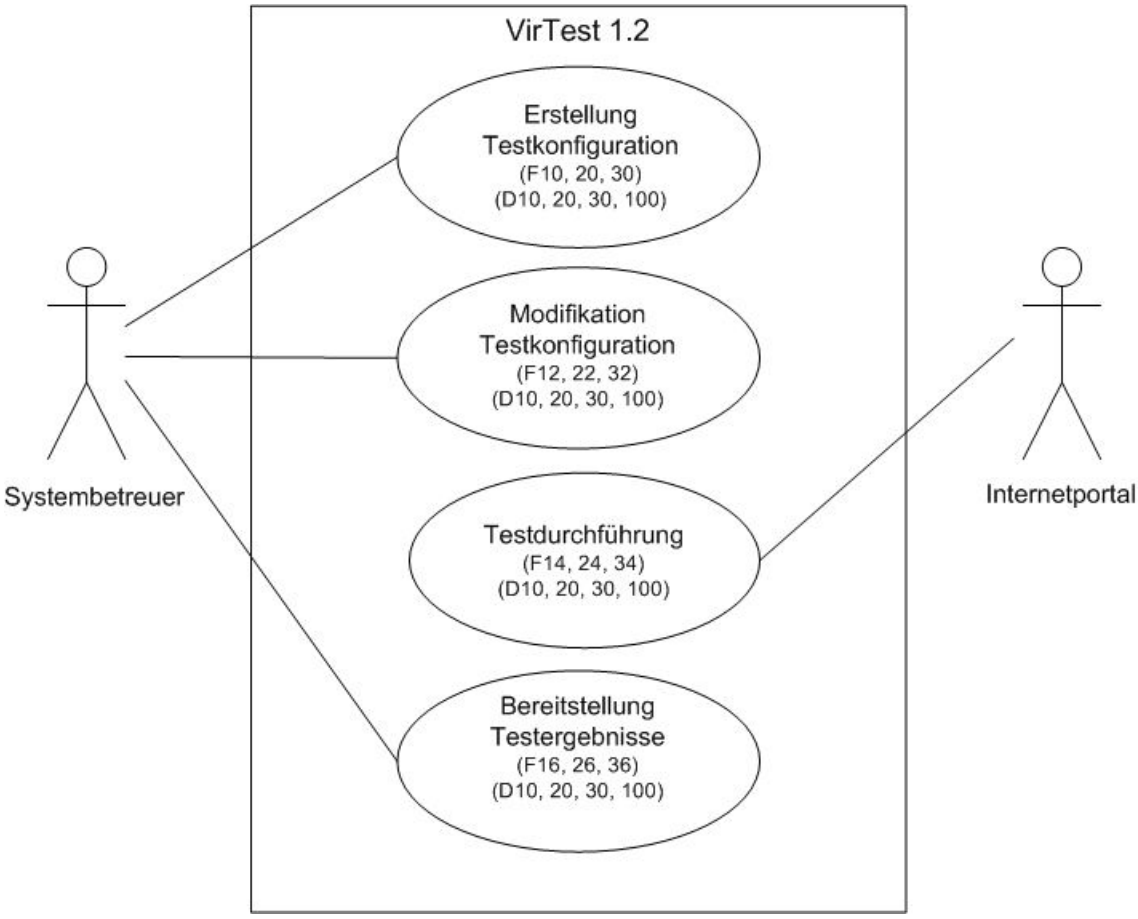
Anwender der Software sind ausschließlich Mitarbeiter, die für den Betrieb des Virtuellen Rathauses verantwortlich sind und mit der Systemumgebung und deren Funktionsweise vertraut sind. Diese werden nachfolgend als Systembetreuer bezeichnet.

A2.3 Betriebsbedingungen

Büroumgebung

A3 Produktübersicht

A3.1 Übersichtsdiagramm



A4 Produktfunktionen

/F10/

Produktfunktion: Erstellung einer neuen Testkonfiguration zur Kontrolle von Weboberflächen

Ziel: Der Systembetreuer erhält eine neue Testkonfiguration mit allen Testschritten

Vorbedingung: Bestehende Datenbankverbindung und Schreibrechte in der Datenbank beziehungsweise in den relevanten Tabellen

Nachbedingung Erfolg: Die Testparameter sind abrufbar in der Datenbank gespeichert

Nachbedingung Fehlschlag: Es konnte keine Testkonfiguration erstellt werden

Akteure: Systembetreuer

Auslösendes Ereignis: Systembetreuer möchte den Funktionstest „Kontrolle von Weboberflächen“ anlegen und speichern

Beschreibung:

1. Es können alle Testkonfigurationen /D10/ vorgenommen werden.

2. Speicherung der Testkonfiguration in der Datenbank.

/F12/

Produktfunktion: Modifikation der Testkonfiguration zur Kontrolle von Weboberflächen

Ziel: Die Modifikationen der Testkonfiguration durch den Systembetreuer werden in der Datenbank gespeichert.

Vorbedingung: Bestehende Datenbankverbindung, Lese- und Schreibrechte in der relevanten Datenbank, Zugriff auf die Produktdaten /D10/

Nachbedingung Erfolg: Testparameter sind abrufbar in der Datenbank gespeichert

Nachbedingung Fehlschlag: Es konnte keine Testkonfiguration modifiziert werden

Akteure: Systembetreuer

Auslösendes Ereignis: Systembetreuer möchte einen Funktionstest modifizieren

Beschreibung:

1. Es wird eine bestehende Testkonfiguration aus der Datenbank gelesen

2. Die Testkonfiguration wird modifiziert und in der Datenbank gespeichert

/F14/

Produktfunktion: Durchführung des Tests „Kontrolle von Weboberflächen“

Ziel: Erfolgreiche Testdurchführung mit Bereitstellung der Testergebnisse und der relevanten Testparameter

Vorbedingung: Bestehende Datenbankverbindung, Leserechte in der relevanten Datenbank, Zugriff auf die Produktdaten /D10/

Nachbedingung Erfolg: Die Testergebnisse und Testparameter werden zur Verfügung gestellt

Nachbedingung Fehlschlag: Es liegen keine Testergebnisse vor

Akteure: Black-Box-Testsystem (VirTest) – Internetangebot (Virtuelles Rathaus)

Auslösendes Ereignis: Der Systembetreuer startet den Test

Beschreibung:

1. Es wird eine bestehende Testkonfiguration aus der Datenbank geladen

2. Der Test wird mittels der Testdaten durchgeführt

3. Bereitstellung der Testergebnisse und der relevanten Testparameter

/F16/

Produktfunktion: Ausgabe der Testergebnisse des Tests „Kontrolle von Weboberflächen

Ziel: Information der Systembetreuer über die Testergebnisse

Vorbedingung: Die aktuellen Testergebnisse wurden ermittelt

Nachbedingung Erfolg: Die Testergebnisse werden im Ausgabefenster dargestellt

Nachbedingung Fehlschlag: Es liegen dem Systembetreuer keine Testergebnisse vor

Akteure: Black-Box-Testsystem (VirTest) - Systembetreuer

Auslösendes Ereignis: Erfolgreiche Testdurchführung

Beschreibung:

1. Die Testergebnisse werden im Ausgabefenster dargestellt

/F20/

Produktfunktion: Erstellung einer neuen Testkonfiguration zur Kontrolle von HTML-Links

Ziel: Der Systembetreuer erhält eine neue Testkonfiguration mit allen Testschritten

Vorbedingung: Bestehende Datenbankverbindung und Schreibrechte in der Datenbank beziehungsweise in den relevanten Tabellen

Nachbedingung Erfolg: Die Testparameter sind abrufbar in der Datenbank gespeichert

Nachbedingung Fehlschlag: Es konnte keine Testkonfiguration erstellt werden

Akteure: Systembetreuer

Auslösendes Ereignis: Systembetreuer möchte den Funktionstest „HTML-Links“ anlegen und speichern

Beschreibung:

1. Es können alle Testkonfigurationen /D20/ vorgenommen werden.

2. Speicherung der Testkonfiguration in der Datenbank.

/F22/

Produktfunktion: Modifikation der Testkonfiguration zur Kontrolle von HTML-Links

Ziel: Die Modifikationen der Testkonfiguration durch den Systembetreuer werden in der Datenbank gespeichert.

Vorbedingung: Bestehende Datenbankverbindung, Lese- und Schreibrechte in der relevanten Datenbank, Zugriff auf die Produktdaten /D20/

Nachbedingung Erfolg: Testparameter sind abrufbar in der Datenbank gespeichert

Nachbedingung Fehlschlag: Es konnte keine Testkonfiguration modifiziert werden

Akteure: Systembetreuer

Auslösendes Ereignis: Systembetreuer möchte einen Funktionstest modifizieren

Beschreibung:

1. Es wird eine bestehende Testkonfiguration aus der Datenbank gelesen

2. Die Testkonfiguration wird modifiziert und in der Datenbank gespeichert

/F24/

Produktfunktion: Durchführung des Tests „HTML-Links“

Ziel: Erfolgreiche Testdurchführung mit Bereitstellung der Testergebnisse und der relevanten Testparameter

Vorbedingung: Bestehende Datenbankverbindung, Leserechte in der relevanten Datenbank, Zugriff auf die Produktdaten /D20/

Nachbedingung Erfolg: Die Testergebnisse und Testparameter werden zur Verfügung gestellt

Nachbedingung Fehlschlag: Es liegen keine Testergebnisse vor

Akteure: Black-Box-Testsystem (VirTest) – Internetangebot (Virtuelles Rathaus)

Auslösendes Ereignis: Der Systembetreuer startet den Test

Beschreibung:

1. Es wird eine bestehende Testkonfiguration aus der Datenbank geladen

2. Der Test wird mittels der Testdaten durchgeführt

3. Bereitstellung der Testergebnisse und der relevanten Testparameter

/F26/

Produktfunktion: Ausgabe der Testergebnisse des Tests „HTML-Links

Ziel: Information der Systembetreuer über die Testergebnisse

Vorbedingung: Die aktuellen Testergebnisse wurden ermittelt

Nachbedingung Erfolg: Die Testergebnisse werden im Ausgabefenster dargestellt

Nachbedingung Fehlschlag: Es liegen dem Systembetreuer keine Testergebnisse vor

Akteure: Black-Box-Testsystem (VirTest) - Systembetreuer

Auslösendes Ereignis: Erfolgreiche Testdurchführung

Beschreibung:

1. Die Testergebnisse werden im Ausgabefenster dargestellt

/F30/

Produktfunktion: Erstellung einer neuen Testkonfiguration zur Kontrolle von Transaktionen

Ziel: Der Systembetreuer erhält eine neue Testkonfiguration mit allen Testschritten

Vorbedingung: Bestehende Datenbankverbindung und Schreibrechte in der Datenbank beziehungsweise in den relevanten Tabellen

Nachbedingung Erfolg: Die Testparameter sind abrufbar in der Datenbank gespeichert

Nachbedingung Fehlschlag: Es konnte keine Testkonfiguration erstellt werden

Akteure: Systembetreuer

Auslösendes Ereignis: Systembetreuer möchte den Funktionstest „Kontrolle von Transaktionen“ anlegen und speichern

Beschreibung:

1. Es können alle Testkonfigurationen /D30/ vorgenommen werden.

2. Speicherung der Testkonfiguration in der Datenbank.

/F32/

Produktfunktion: Modifikation der Testkonfiguration zur Kontrolle von Transaktionen

Ziel: Die Modifikationen der Testkonfiguration durch den Systembetreuer werden in der Datenbank gespeichert.

Vorbedingung: Bestehende Datenbankverbindung, Lese- und Schreibrechte in der relevanten Datenbank, Zugriff auf die Produktdaten /D30/

Nachbedingung Erfolg: Testparameter sind abrufbar in der Datenbank gespeichert

Nachbedingung Fehlschlag: Es konnte keine Testkonfiguration modifiziert werden

Akteure: Systembetreuer

Auslösendes Ereignis: Systembetreuer möchte einen Funktionstest modifizieren

Beschreibung:

1. Es wird eine bestehende Testkonfiguration aus der Datenbank gelesen

2. Die Testkonfiguration wird modifiziert und in der Datenbank gespeichert

/F34/

Produktfunktion: Durchführung des Tests „Kontrolle von Transaktionen“

Ziel: Erfolgreiche Testdurchführung mit Bereitstellung der Testergebnisse und der relevanten Testparameter

Vorbedingung: Bestehende Datenbankverbindung, Leserechte in der relevanten Datenbank, Zugriff auf die Produktdaten /D30/

Nachbedingung Erfolg: Die Testergebnisse und Testparameter werden zur Verfügung gestellt

Nachbedingung Fehlschlag: Es liegen keine Testergebnisse vor

Akteure: Black-Box-Testsystem (VirTest) – Internetangebot (Virtuelles Rathaus)

Auslösendes Ereignis: Der Systembetreuer startet den Test

Beschreibung:

1. Es wird eine bestehende Testkonfiguration aus der Datenbank geladen

2. Der Test wird mittels der Testdaten durchgeführt

3. Bereitstellung der Testergebnisse und der relevanten Testparameter

/F36/

Produktfunktion: Ausgabe der Testergebnisse des Tests „Kontrolle von Transaktionen"

Ziel: Information der Systembetreuer über die Testergebnisse

Vorbedingung: Die aktuellen Testergebnisse wurden ermittelt

Nachbedingung Erfolg: Die Testergebnisse werden im Ausgabefenster dargestellt

Nachbedingung Fehlschlag: Es liegen dem Systembetreuer keine Testergebnisse vor

Akteure: Black-Box-Testsystem (VirTest) - Systembetreuer

Auslösendes Ereignis: Erfolgreiche Testdurchführung

Beschreibung:

1. Die Testergebnisse werden im Ausgabefenster dargestellt

A5 Produktdaten

Die Parameter zur Testdurchführung werden in einer MySQL-Datenbank abgelegt, deren genaue Struktur noch zu spezifizieren ist. Die Ausgabedaten werden im Ausgabefenster dargestellt.

Eine grobe Struktur liefern die folgenden Daten:

/D10/ Testkonfiguration (Kontrolle von Weboberflächen) max. 100:

ID (fortlaufend vom System), Testname, Testbeschreibung, Testschritt-Nr., URL-Pfad, Suchbegriff, Timeout, Testschrittbeschreibung, HTTP-Error-Codes, HTTP-Error-Messages

/D20/ Testkonfiguration (Kontrolle von HTML-Links) max. 100:

ID (fortlaufend vom System), Testname, Testbeschreibung, Testschritt-Nr., URL-Pfad, Timeout, Testschrittbeschreibung, HTTP-Error-Codes, HTTP-Error-Messages

/D30/ Testkonfiguration (Kontrolle von Transaktionen) max. 100:

ID (fortlaufend vom System), Testname, Testbeschreibung, Testschritt-Nr., URL-Pfad, Übergabeparameter (nach dem URL-Pfad), Suchbegriff, Timeout, Testschrittbeschreibung, HTTP-Error-Codes, HTTP-Error-Messages

/D100/ Mailkonfiguration (max. 1 pro Testgruppe)

Mailserver, Mailsender, Mailempfänger, Betreff

A6 Produktleistungen

Hinsichtlich der Produktleistungen wurden vom HABIT keine Vorgaben gemacht.

A7 Software-Qualitätsmerkmale

Hinsichtlich der Produktleistungen wurden vom HABIT keine Vorgaben gemacht.

A8 Benutzungsoberfläche

Für die Einstellung der Testkonfigurationen /D10/ bis /D30/ und der Mailkonfiguration /D100/ sollte eine grafische Benutzungsoberfläche entwickelt werden. Mit dieser Benutzungsoberfläche, müsste es möglich sein, speziell den URL-Pfad und die Übergabeparameter (z.B. aus /D10/) aus der Zwischenablage (z.B. bei einem Web-Browser) zu übernehmen. Eine Benutzungsoberfläche, die mit einem Web-Browser erzeugt wird ist ebenfalls möglich.

Standardmäßig ist eine menüorientierte Bedienung vorzusehen, die auf eine Mausbedienung ausgelegt ist.

A9 Technische Produktumgebung

A9.1 Software

Betriebssystem: Windows NT und Windows 2000

Programmiersprachen: Perl Version 5.6.1 und Zusatzmodule

DBMS: MySQL

A9.2 Hardware

Standard Office PC

A9.3 Orgware

Netzwerkverbindung zum Internet und/oder zum Lokalen Netzwerk (LAN)

A10 Spezielle Anforderungen an die Entwicklungsumgebung

Der HABIT wünscht eine Umsetzung der Anwendungssoftware mit einer kostenlosen bzw. mit einer bereits vorhandenen Entwicklungssoftware.

A11 Versionsinformationen

Version	Autor	Datum	Status	Kommentar
1.0	Andreas Zarfl	01.08.2004	akzeptiert	Erstellung der Grundversion des Pflichtenhefts
1.1	Andreas Zarfl	15.09.2004	akzeptiert	Notwendige Anpassungen zur Datenhaltung der Testdaten in einer MySQL-Datenbank
1.2	Andreas Zarfl	30.01.2005	akzeptiert	Der Funktionstest zur Kontrolle eines Verzeichnisses auf einem bestimmten Server wird nicht mehr benötigt. Integration der Funktion „Basis-URL-Prüfung“ Vollständige Überarbeitung des Pflichtenheftes.

Anhang B SQL-Quellcode

B1.1 Übersicht über die Tabellenstrukturen

Konfiguration_Linkueberpruefung

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
Testschritt_Id	INTEGER		NN	UNSIGNED			
Testschritt_Testschrittnummer	SMALLINT		NN	UNSIGNED			
URL_Pfad	VARCHAR(250)		NN				
Timeout	SMALLINT		NN	UNSIGNED	1		

IndexName	IndexType	Columns
Konfiguration_Linkueberpruefung_FKIndex1	Index	Testschritt_Testschrittnummer Testschritt_Id

Konfiguration_Oberflaechenkontrolle

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
Testschritt_Id	INTEGER		NN	UNSIGNED			
Testschritt_Testschrittnummer	SMALLINT		NN	UNSIGNED			
URL_Pfad	VARCHAR(250)		NN				
Suchbegriff	VARCHAR(50)		NN				
Timeout	SMALLINT		NN	UNSIGNED	1		

IndexName	IndexType	Columns
Testschrittkonfiguration_Weboberflächenkontrolle_FKIndex1	Index	Testschritt_Testschrittnummer Testschritt_Id

Konfiguration_Transaktionskontrolle

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
Testschritt_Id	INTEGER		NN	UNSIGNED			
Testschritt_Testschrittnummer	SMALLINT		NN	UNSIGNED			
URL_Pfad	VARCHAR(250)		NN				
Uebergabewert	VARCHAR(250)						
Suchbegriff	VARCHAR(50)						
Timeout	SMALLINT		NN	UNSIGNED	1		

IndexName	IndexType	Columns
Testschrittkonfiguration_Verzeichniskontrolle_FKIndex1	Index	Testschritt_Testschrittnummer Testschritt_Id

Mail

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
Mail_Id	**INTEGER**	PK	NN				AI
Testgruppe_Testgruppenname	VARCHAR(100)		NN				
Mailserver	VARCHAR(100)						
Mailsender	VARCHAR(100)						
Mailempfaenger	VARCHAR(100)						
Betreff	VARCHAR(200)						

IndexName	IndexType	Columns
PRIMARY	PRIMARY	Mail_Id
Table_09_FKIndex1	Index	Testgruppe_Testgruppenname

Testfall

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
Testfallname	**VARCHAR(100)**	PK	NN				
Testgruppe_Testgruppenname	VARCHAR(100)		NN				
Testfallbeschreibung	VARCHAR(250)						

IndexName	IndexType	Columns
PRIMARY	PRIMARY	Testfallname
Testfall_FKIndex1	Index	Testgruppe_Testgruppenname

Testgruppe

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
Testgruppenname	**VARCHAR(100)**	PK	NN				
Testgruppenbeschreibung	VARCHAR(250)						

IndexName	IndexType	Columns
PRIMARY	PRIMARY	Testgruppenname

Testschritt

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
Testschritt_Id	**INTEGER**	PK	NN	UNSIGNED			AI
Testschrittnummer	**SMALLINT**	PK	NN	UNSIGNED			
Testfall_Testfallname	VARCHAR(100)		NN				
Testtyp_idTesttyp	SMALLINT		NN	UNSIGNED			
Testschrittbeschreibung	VARCHAR(250)						

IndexName	IndexType	Columns
PRIMARY	PRIMARY	Testschritt_Id Testschrittnummer
Testschritt_FKIndex1	Index	Testfall_Testfallname
Testschritt_FKIndex2	Index	Testtyp_idTesttyp

Testtyp

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
idTesttyp	**SMALLINT**	PK	NN	UNSIGNED			AI
Bezeichnung	VARCHAR(50)						

IndexName	IndexType	Columns
PRIMARY	PRIMARY	idTesttyp

B1.2 SQL-Anweisungen zur Erzeugung der Tabellen

```
CREATE TABLE Testgruppe (
  Testgruppenname VARCHAR(100) NOT NULL,
  Testgruppenbeschreibung VARCHAR(250) NULL,
  PRIMARY KEY(Testgruppenname)
)


CREATE TABLE Testtyp (
  idTesttyp SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
  Bezeichnung VARCHAR(50) NULL,
  PRIMARY KEY(idTesttyp)
)


CREATE TABLE Testfall (
  Testfallname VARCHAR(100) NOT NULL,
  Testgruppe_Testgruppenname VARCHAR(100) NOT NULL,
  Testfallbeschreibung VARCHAR(250) NULL,
  PRIMARY KEY(Testfallname),
  INDEX Testfall_FKIndex1(Testgruppe_Testgruppenname),
  FOREIGN KEY(Testgruppe_Testgruppenname)
    REFERENCES Testgruppe(Testgruppenname)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION
)


CREATE TABLE Mail (
  Mail_Id INTEGER NOT NULL AUTO_INCREMENT,
  Testgruppe_Testgruppenname VARCHAR(100) NOT NULL,
  Mailserver VARCHAR(100) NULL,
  Mailsender VARCHAR(100) NULL,
  Mailempfaenger VARCHAR(100) NULL,
  Betreff VARCHAR(200) NULL,
  PRIMARY KEY(Mail_Id),
  INDEX Table_09_FKIndex1(Testgruppe_Testgruppenname),
  FOREIGN KEY(Testgruppe_Testgruppenname)
    REFERENCES Testgruppe(Testgruppenname)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION
)
```

```
CREATE TABLE Testschritt (
  Testschritt_Id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  Testschrittnummer SMALLINT UNSIGNED NOT NULL,
  Testfall_Testfallname VARCHAR(100) NOT NULL,
  Testtyp_idTesttyp SMALLINT UNSIGNED NOT NULL,
  Testschrittbeschreibung VARCHAR(250) NULL,
  PRIMARY KEY(Testschritt_Id, Testschrittnummer),
  INDEX Testschritt_FKIndex1(Testfall_Testfallname),
  INDEX Testschritt_FKIndex2(Testtyp_idTesttyp),
  FOREIGN KEY(Testfall_Testfallname)
    REFERENCES Testfall(Testfallname)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  FOREIGN KEY(Testtyp_idTesttyp)
    REFERENCES Testtyp(idTesttyp)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION
)


CREATE TABLE Konfiguration_Linkueberpruefung (
  Testschritt_Id INTEGER UNSIGNED NOT NULL,
  Testschritt_Testschrittnummer SMALLINT UNSIGNED NOT NULL,
  URL_Pfad VARCHAR(250) NOT NULL,
  Timeout SMALLINT UNSIGNED NOT NULL DEFAULT 1,
  INDEX Konfiguration_Linkueberpruefung_FKIndex1(Testschritt_Testschrittnummer,
Testschritt_Id),
  FOREIGN KEY(Testschritt_Testschrittnummer, Testschritt_Id)
    REFERENCES Testschritt(Testschrittnummer, Testschritt_Id)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION
)


CREATE TABLE Konfiguration_Oberflaechenkontrolle (
  Testschritt_Id INTEGER UNSIGNED NOT NULL,
  Testschritt_Testschrittnummer SMALLINT UNSIGNED NOT NULL,
  URL_Pfad VARCHAR(250) NOT NULL,
  Suchbegriff VARCHAR(50) NOT NULL,
  Timeout SMALLINT UNSIGNED NOT NULL DEFAULT 1,
  INDEX Testschrittkonfigurati-
on_Weboberflächenkontrolle_FKIndex1(Testschritt_Testschrittnummer, Test-
schritt_Id),
  FOREIGN KEY(Testschritt_Testschrittnummer, Testschritt_Id)
    REFERENCES Testschritt(Testschrittnummer, Testschritt_Id)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION
)
```

```
CREATE TABLE Konfiguration_Transaktionskontrolle (
  Testschritt_Id INTEGER UNSIGNED NOT NULL,
  Testschritt_Testschrittnummer SMALLINT UNSIGNED NOT NULL,
  URL_Pfad VARCHAR(250) NOT NULL,
  Uebergabewert VARCHAR(250) NULL,
  Suchbegriff VARCHAR(50) NULL,
  Timeout SMALLINT UNSIGNED NOT NULL DEFAULT 1,
  INDEX Testschrittkonfigurati-
on_Verzeichniskontrolle_FKIndex1(Testschritt_Testschrittnummer, Testschritt_Id),
  FOREIGN KEY(Testschritt_Testschrittnummer, Testschritt_Id)
    REFERENCES Testschritt(Testschrittnummer, Testschritt_Id)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION
)
```

Anhang C Inhalt und Nutzung der CD-ROM

Auf der mitgelieferten CD-ROM befindet sich die gesamte Dokumentation der Diplomarbeit.

Im Einzelnen sind dies:

- Die Dokumentation der Diplomarbeit im PDF-Format
- Die Installationsdateien für das RDBMS MySQL 4.0 und den Windows Perl Interpreter von ActiveState (Version 5.6.1)
- Vollständiger Quellcode der Software VirTest (Version 1.2) inklusive relevanter Module
- Perl-Skript zur Erzeugung des Datenbankschemas

Zur Installation der Software wurde eine HTML-Datei („Installationsanleitung von VirTest Version 1_2.htm“) angelegt, in der die notwendigen Installationsschritte angegeben sind.

Anhang D Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig angefertigt habe. Ich habe mich keiner fremden Hilfe bedient und keine anderen, als die angegebenen Quellen und Hilfsmittel benutzt. Alle Stellen, die wörtlich oder sinngemäß veröffentlichten oder nicht veröffentlichten Schriften und anderen Quellen entnommen sind, habe ich als solche kenntlich gemacht. Diese Arbeit hat in gleichen oder wesentlichen Teilen noch keiner Prüfungsbehörde vorgelegen.

Bottrop, den 03.04.2005 _____

(Vorname Nachname)