Universität Leipzig
Fakultät für Mathematik und Informatik
Institut für Informatik

Konzeption und Implementierung eines Anzeigenannahme-, Service- und Verwaltungssystems für das Publizieren und Verwalten von gestalteten Inseraten für Anzeigen-Online-Dienste im World Wide Web

DIPLOMARBEIT

Leipzig, August 1999

vorgelegt von
Christian Blümel

Danksagung

Herrn Prof. Dr. Gerhard Heyer danke ich für die Vergabe der Aufgabenstellung und die Betreuung der Diplomarbeit.

Inhaltsverzeichnis

Abbildungsverzeichnis

I. Einleitung

1. Anzeigen-Online-Dienste

Anzeigen-Online-Dienste wie Kleinanzeigendienste, Stellen- und Immobilienbörsen erfreuen sich im *World Wide Web* (WWW), dem multimedialen Dienst des Internets, großer Beliebtheit: Bei *Dino-Online*[1], einem der führenden WWW-Kataloge in Deutschland, sind allein über 100 Stellen- und Immobilienbörsen und mehr als 130 überregionale Kleinanzeigendienste für Deutschland verzeichnet. Beispiele für Kleinanzeigendienste sind *Das Schwarze Brett im Internet*[2], ein Anzeigendienst von und für Studenten, sowie die kommerziellen Dienste *Webm@rkt*[3] und *IQ*[4].

1.1 Vergleich Anzeigendienst – Homepage

Gegenüber der Veröffentlichung einer Anzeige auf einer privaten *Homepage* oder der Homepage eines Unternehmens haben Anzeigendienste entscheidende Vorteile:

1. Vorteil: Bekanntheitsgrad
   Während die Adresse (*Uniform Ressource Locator*, URL) der Homepage eines einzelnen Internetteilnehmers und vor allem auch von vielen kleineren Unternehmen kaum einem breiten Internet-Teilnehmerkreis bekannt ist, können etablierte Anzeigendienste nach eigenen Angaben auf mehrere tausend Besucher pro Monat verweisen.

2. Vorteil: Aktualität
   Bei bekannten Anzeigendiensten werden täglich neue Anzeigen geschaltet, die die Internetnutzer animieren, mehrmals pro Woche oder sogar täglich nach interessanten Angeboten zu suchen. So ist die Wahrscheinlichkeit sehr groß, daß eine neugeschaltete Anzeige von vielen Internetnutzern gelesen wird. Private Homepages oder die Homepages von Unternehmen werden dagegen sehr viel seltener mit neuen Inhalten versehen, so daß der Anreiz, diese Seiten häufiger zu besuchen, kleiner ist.

3. Vorteil: Strukturierung der Anzeigen und eine einheitliche Oberfläche
   Anzeigendienste bieten dem Internetnutzer die geschalteten Inserate nach verschiedenen Kriterien geordnet an wie zum Beispiel Rubriken, Schaltdatum, Preis, m²-Zahl. So sind die einzelnen Anzeigen für den Nutzer direkt und schnell zugänglich. Langes Suchen in WWW-Katalogen oder Suchmaschinen entfällt. Der geringere Zeitaufwand und die damit gesparten Telefonkosten sind entscheidende Vorteile gegenüber der Suche nach Anzeigen auf einzelnen privaten Homepages oder den Homepages einzelner Unternehmen.

---

[1] http://www.dino-online.de

[2] http://dsb.uni-leipzig.de

[3] http://www.webmarkt.de

[4] http://www.iqanzeigen.de

Hinzu kommt, daß sich der Nutzer auf jeder Homepage neu orientieren muß, da jede Homepage anders gegliedert und gestaltet ist. Auch dies erfordert Zeit. Anders bei Anzeigendiensten: Hier sind die Anzeigen unter einer einheitlichen Oberfläche zugänglich. Nach einer einmaligen Orientierung beim ersten Besuch findet der Nutzer beim nächsten Besuch direkt zu den gesuchten Anzeigen.

4. Vorteil: Betrieb eines eigenen Servers ist nicht erforderlich
   Ohne Anzeigendienste müßten Privatpersonen oder Unternehmen, die eigene Inhalte wie Klein-, Produkt-, Dienstleistungs- oder Stellenanzeigen im WWW veröffentlichen möchten, einen eigenen Server mit Standleitung ins Internet einrichten bzw. Platz auf einem Server mieten.

Allerdings haben derzeit verfügbare Anzeigen-Online-Dienste auch Nachteile gegenüber einer eigenen Homepage:

1. Nachteil: Gestaltungmöglichkeiten der Inserate
   Im Vergleich zu einer eigenen Homepage bieten derzeit verfügbare Kleinanzeigendienste dem Inserenten nur sehr beschränkte Gestaltungsmöglichkeiten seiner Anzeige. So bestehen die Kleinanzeigen aus einer Überschrift, einem Textkörper und meist noch der Angabe einer Email-Adresse, um Kontakt mit dem Inserenten aufnehmen zu können. Einen anderen Ansatz verfolgen Anzeigendienste wie die Stellenbörse von *Jobs & Adverts*[5], die gestaltete Anzeigen in Form von *HTML*-Seiten veröffentlichen, wobei auch Grafiken und Bilder eingebunden werden können.

   Allerdings erfährt der Inserent hier große Einschränkungen in der Handhabung der Anzeige:

2. Nachteil: Handhabung der Anzeigen
   Während viele Kleinanzeigendienste die komfortable Aufgabe der Anzeige über ein WWW-Formular ermöglichen, besteht nach dem Freischalten der Anzeige für den Inserenten meist nur noch die Möglichkeit, die Anzeige zu löschen oder zu verlängern.
   Im Gegensatz dazu bietet eine eigene Homepage die Möglichkeit, den Inhalt der Anzeigen ständig zu verändern. Beim Anzeigendienst *Jobs & Adverts* können zwar gestaltete Inserate veröffentlicht werden, allerdings können diese Anzeigen nicht über das WWW aufgegeben werden, sondern müssen per Email oder Diskette bzw. als Vorlage zum Einscannen eingesandt werden. Das Unternehmen, das diesen Anzeigendienst betreibt, muß nun "per Hand" die zu einer Präsentation gehörigen Dateien entgegennehmen (per Email, auf Diskette); die Dateien in bestimmte Verzeichnisse kopieren und die Anzeigendaten (Auftrags- bzw. Referenznummer, Rubrik, Schaltbeginn, Schaltdauer etc.) in eine Datenbank eingeben. Dieses Verfahren erfordert Zeit und ist bei einer größeren Anzahl von Anzeigen nur mit einer entsprechend großen Anzahl von Mitarbeitern zu realisieren.

[5] http://www.jobpilot.de

2. Aufgabenstellung

Bestehende Anzeigen-Online-Dienste im WWW weisen entweder Einschränkungen bei der Gestaltungsmöglichkeit der Inserate auf oder bieten keine Annahme- und nachträgliche Veränderungsmöglichkeit gestalteter Anzeigen über das WWW an. Betreiber von Anzeigen-Online-Diensten, die ihren Kunden das Schalten gestalteter Anzeigen ermöglichen wollen, nehmen bisher einen großen zeitlichen und damit finanziellen Aufwand für die Verwaltung dieser Anzeigen in Kauf.

Aufgabenstellung dieser Diplomarbeit ist die Konzeption und Entwicklung eines *Anzeigenannahme-, Service- und Verwaltungssystems* (ASV-System) für das Inserieren und Verwalten gestalteter Anzeigen im World Wide Web, das eine Lösungsmöglichkeit für diese Einschränkungen und Problemstellungen aufzeigt.

3. Hauptforderungen

3.1 Aufstellung der Hauptforderungen

An das ASV-System sind damit folgende Hauptforderungen hinsichtlich Funktionalität und technischer Voraussetzungen zu stellen, die im Anschluß an die Aufstellung diskutiert werden:

Der Nutzer, der die Inserate liest, sollte:

1. Hauptforderung: sich die Anzeigen über das WWW - nur unter Benutzung eines üblichen WWW-Browsers - nach verschiedenen Kriterien geordnet anzeigen lassen können.

Der Inserent des Anzeigendienstes (im folgenden *Kunde* genannt) sollte:

2. Hauptforderung: die von ihm gestalteten Inserate direkt über das WWW - nur unter Benutzung eines üblichen WWW-Browsers - zum Anzeigendienst transferieren, per Voransicht kontrollieren und dabei auch alle Anzeigendaten (gewünschte Rubrik, Schaltbeginn, Schaltdauer etc.) angeben können (Anzeigenannahme),
3. Hauptforderung: während der gesamten Schaltdauer jederzeit die Kontrolle über die Anzeige behalten, d.h., daß er das Inserat zu jedem Zeitpunkt verändern können sollte, indem er zum Beispiel Dateien hinzufügt bzw. aktualisiert oder löscht (Service) - wiederum nur unter Benutzung eines üblichen WWW-Browsers.

Der Betreiber des Anzeigendienstes sollte:

4. Hauptforderung: den Anzeigendienst über das WWW - nur unter Benutzung eines üblichen WWW-Browsers - betreiben können (Verwaltung).

3.2 Diskussion der Hauptforderungen

Die erste Hauptforderung ist entscheidend für die Akzeptanz des Dienstes: Der Internetnutzer muß sich die Inserate auf die im WWW übliche Art und Weise, d.h. unter Benutzung eines WWW-Browsers, betrachten können.
Das bedeutet, daß die Anzeigen in Form der im WWW üblichen HTML-Dateien mit eingebetteten *GIF*-Grafiken, *JPEG*-Bildern, *Java*-Applets etc. vorliegen müssen. Darüber hinaus sollte der Kunde die Möglichkeit haben, weitere Dateien für den Nutzer zum Download anzubieten. Diese Forderung bedeutet auch, daß der Kunde kein spezielles Programm benötigt, das die Anzeigen in einem proprietären Format abspeichert, sondern seine Anzeigen in der im WWW üblichen Weise in Form von HTML-Seiten erstellen kann, was mit neueren Text- und Grafikprogrammen problemlos möglich ist.

Die zweite und dritte Hauptforderung stellen die Anforderungen hinsichtlich der Funktionaliät für den Kunden dar. Der wichtigste Punkt: Der Kunde soll in der Lage sein, seine gestalteten Anzeigen nur unter Benutzung seines WWW-Browsers zum Anzeigendienst zu transferieren und dabei auch die gewünschten Anzeigendaten zu bestimmen. Da keine zusätzlichen Programme für das Inserieren benötigt werden, die erst beschafft und installiert werden müssen, und die Vorgehensweise der bei Kleinanzeigendiensten gleicht, ist eine hohe Akzeptanz seitens des Kundens gewährleistet. Die zweite Hauptforderung impliziert auch, daß die Anzeigen nicht erst auf einem WWW-Server des Kunden bereitgestellt werden müssen, um dann auf den WWW-Server des Anzeigendienstes übertragen werden zu können.

Gegenüber bestehenden Anzeigendiensten, die die Zusendung der multimedial gestalteten Inserate per Diskette, Email etc. verlangen, stellen die mit der zweiten und dritten Hauptforderung erzielbaren Verbesserungen einen entscheidenden Fortschritt dar:

- Die Zeit, bis ein Inserat erscheinen kann, verkürzt sich erheblich, auch im Vergleich zur Zusendung per Email, da die Anzeige nach der Annahme bereits anzeigefertig und vom Inserenten kontrolliert (per Voransicht) auf dem Server des Anzeigendienstes vorliegt, weitere Bearbeitungen entfallen und auch die Anzeigendaten (wie gewünschte Rubrik, Schaltbeginn etc.) in eindeutiger Form aufgenommen wurden.
- Das Konvertieren von Diskettenformaten sowohl für den Kunden als auch für den Betreiber des Anzeigendienstes entfällt, da die Dateien direkt auf dem Zielrechner, dem Server des Anzeigendienstes, gespeichert werden: Dem Kunden muß man nicht vorschreiben, daß er auf seinem *Apple*-Rechner die zur Anzeige gehörenden Dateien auf PC-formatierten Disketten speichern muß, andererseits benötigt der Betreiber keinen Apple-Rechner oder spezielle Programme, die Apple-Disketten lesen können, wenn der Kunde einmal eine Apple-Diskette zugesandt hat.
- Für den Kunden ermöglicht die geforderte Voransichtfunktion eine genaue Kontrolle des Ergebnisses. Er sieht schon bei der Anzeigenannahme, wie sein Inserat später für den Nutzer erscheint.

- Durch die gleichzeitige und eindeutige Aufnahme von Anzeige und Anzeigendaten (Rubrik, Schaltbeginn, Schaltdauer etc.) werden Mißverständnisse vermieden, die etwa bei der telefonischen Übermittlung der gewünschten Anzeigendaten entstehen können.
- Die Möglichkeit, bereits geschaltete Inserate jederzeit inhaltlich zu verändern, ermöglicht dem Kunden, seine Anzeige wie auf einer eigenen Homepage zu behandeln. Für den Betreiber des Anzeigendienstes stellt diese Möglichkeit ein zusätzliches Funktionsangebot dar, das die Attraktivität des Dienstes erhöht und bei kommerziell betriebenen Diensten zusätzlich abgerechnet werden kann.
- Für den Betreiber bedeutet die gleichzeitige Aufnahme von anzeigefertigem Inserat und Anzeigendaten, daß sich der Betrieb des Anzeigendienstes weitestgehend automatisieren und die Verwaltungstätigkeit im wesentlichen auf eine Kontroll- und Abrechnungsfunktion reduzieren läßt, sowohl bei der Anzeigenannahme als auch bei gewünschten Veränderungen.

Die vierte Hauptforderung, Verwaltung des Anzeigendienstes über einen WWW-Browser, bedeutet für den Betreiber des Anzeigendienstes:

- Die Verwaltung kann *auf* jedem Computer mit Internetanschluß durchgeführt werden, auf dem ein WWW-Browser zur Verfügung steht. Da WWW-Browser heute für nahezu alle Betriebssysteme zur Verfügung stehen, ist der Betreiber des Anzeigendienstes unabhängig von einem bestimmten Betriebssystem wie zum Beispiel *Windows 95, Mac OS, Unix*.
- Die Verwaltung kann *von* jedem Computer mit Internetanschluß aus durchgeführt werden, da das Verwaltungsprogramm nicht als ausführbare Datei auf dem Rechner vorliegen muß, sondern bei Bedarf über den WWW-Browser zur Verfügung steht. Wird der Anzeigendienst von mehreren Mitarbeitern verwaltet, entstehen so keine Probleme mit Programmupdates, da die neueste Version des Verwaltungsprogramms immer über das WWW zur Verfügung steht. Zudem ist es so möglich, daß die Mitarbeiter die eingegangenen Anzeigenaufträge zum Beispiel als Telearbeiter oder von verschiedenen Unternehmensstandorten aus bearbeiten.

Die potentiellen Sicherheitsrisiken, die durch die Verlagerung der Annahme, des Services und Verwaltung ins WWW entstehen, müssen bei der Konzeption berücksichtigt werden.

4. Zielsetzung

Die Ziele dieser Diplomarbeit sind:

1. Die Konzeption eines Anzeigenannahme-, Service- und Verwaltungssystems (ASV-System), das den genannten vier Hauptforderungen entspricht und damit als Grundlage für Anzeigendienste im WWW verwendet werden kann, die das Schalten von gestalteten Anzeigen gestatten.
2. Die Implementierung eines Beispiel-ASV-Systems auf der Grundlage dieser Konzeption.

Dabei liegt der Schwerpunkt auf der Handhabung der gestalteten Anzeigen bei der Annahme, beim Service und in der Verwaltung. Deshalb werden auch folgende Funktionen nicht betrachtet, die bei bestehenden Anzeigendiensten bereits gut

gelöst sind oder beim Aufbau eines kommerziellen Anzeigendienstes sinnvoll wären, aber nicht in direktem Zusammenhang mit dem Schwerpunkt der Arbeit stehen:

- Benutzerverwaltung,
- Buchführung,
- Anzeigenstatistik.

5. Vorgehensweise

Die Entwicklung des ASV-Systems erfolgte nach den in [BA96] beschriebenen Prinzipien und Verfahren des *Software-Engineerings*. Die einzelnen Phasen des Softwareentwicklungsprozesses werden in Kapitel II der Diplomarbeit dargestellt.
Die Entwicklung des ASV-System beginnt mit der Planungsphase in Abschnitt II.1. Hier werden die Hauptanforderungen analysiert und im Ergebnisdokument, dem sogenannten Lastenheft, festgehalten. Ebenso findet in der Planungsphase eine Diskussion zur Durchführbarkeit des Projektes verbunden mit der Auswahl einer geeigneten Technologie für die Implementierung statt.
Die sich anschließende Definitionsphase in Abschnitt II.2 führt über die Anfertigung eines Pflichtenheftes zum Produktmodell, das eine Lösung der sich ergebenden fachlichen Problemstellungen darstellt. Pflichtenheft und Produktmodell haben Auswirkungen auf die Konzeption der Benutzeroberfläche, die ebenfalls in der Definitionsphase angefertigt wird.

Weiterhin wird der Entwicklungsprozeß in der Entwurfsphase, Abschnitt II.3., und Implementierungsphase, Abschnitt II.4., dargestellt. Der Schwerpunkt der Entwurfsphase liegt beim Datenbankentwurf, aber auch Sicherheitsaspekte der Anwendung werden ausführlich behandelt. Die sich im Entwicklungsprozeß anschließende Abnahme- und Einführungsphase wird in Abschnitt II.5 durch eine mit Screenshots illustrierte Ergebnisdokumentation der Beispielrealisierung ersetzt.

Kapitel III enthält die Diskussion des erreichten Entwicklungsstandes und Ausblicke auf eine mögliche Weiterentwicklung des ASV-Systems. Den Abschluß bildet die Zusammenfassung in Kapitel IV.

Die Ergebnisdokumente der einzelnen Phasen der Software-Entwicklung wie Lastenheft, Pflichtenheft, Produktmodell etc. sind in Anhang A bis D dargestellt.

II. Entwicklung des ASV-Systems

1. Planung

Nach [BA96, S.56ff.] steht vor der eigentlichen Entwicklung des Software-Produktes die Erarbeitung einer *Durchführbarkeitsstudie* (*feasibility study*), die die sachliche Grundlage für die Entscheidung liefert, ob ein Produkt entwickelt werden soll oder nicht. Die Durchführbarkeitsstudie besteht aus folgenden Teildokumenten:

- Lastenheft,
- Projektkalkulation,
- Projektplan.

Diese erhält man u.a. durch folgende Einzelaktivitäten:

- Auswählen des Produktes (Trendstudien, Marktanalysen, Kundenanfragen),
- Voruntersuchung des Produktes (Ist-Analyse, Festlegen der Hauptanforderungen),
- Durchführbarkeitsuntersuchung (Prüfen der fachlichen Durchführbarkeit, Prüfen alternativer Lösungsvorschläge, Prüfen der personellen Durchführbarkeit),
- Prüfen der ökonomischen Durchführbarkeit (Aufwands- und Terminschätzung, Wirtschaftlichkeitsrechnung).

Da die *Auswahl des Produktes* durch die Aufgabenstellung der Diplomarbeit gegeben ist, eine *Ist-Analyse* für die relevanten Funktionen bereits in der Einleitung aufgeführt wurde und ökonomische Aspekte der Durchführbarkeitsuntersuchung im Rahmen dieser Diplomarbeit nicht relevant sind, werde ich mich im folgenden auf die fachlichen Aspekte *Festlegen der Hauptanforderungen*, *Prüfen der fachlichen Durchführbarkeit* und das fachliche Ergebnisdokument, das Lastenheft, beschränken.

1.1 Hauptanforderungen an das ASV-System

1.1.1 Hauptfunktionen

Aus den in der Einleitung beschriebenen und diskutierten vier Hauptforderungen ergeben sich folgende Hauptfunktionen des ASV-Systems:

- Aufgabe neuer Anzeigen mit Dateitransfer, Voransicht, Erfassung von Anzeigendaten;
- Verändern, Aussetzen von Anzeigen;
- Freischalten, Verwerfen (Löschen), Zurückstellen neuer Anzeigen;
- Freischalten, Verwerfen (Löschen), Zurückstellen von Veränderungsaufträgen.

1.1.2 Hauptdaten

Folgende Anzeigendaten sollen permanent gespeichert werden:

Anzeigendaten	**Funktion**
Überschrift der Anzeige	für die Darstellung des Anzeigenbestandes im Frontend; als Orientierungsmöglichkeit für den Nutzer, der den Anzeigenbestand betrachtet
Schlagwörter	für die Suche nach Schlagwörtern im Anzeigenbestand
Kurzbeschreibung	für die schnelle Anzeigenauswahl durch den Nutzer in der Anzeigenübersicht
Kontakt-Email-Adresse	für den Kontakt Nutzer - Kunde
Rubrik	für die Strukturierung des Anzeigenbestandes bei der Darstellung
Unterrubrik	für die Strukturierung des Anzeigenbestandes in der Rubrik
Stadt	für die Strukturierung des Anzeigenbestandes in der Unterrubrik
Schaltbeginn	für den automatisierten Betrieb des Anzeigen-dienstes
Schaltende	für den automatisierten Betrieb des Anzeigen-dienstes
Start-HTML-Seite der Anzeige	zur eindeutigen Darstellung der Anzeige bei Inseraten mit mehreren HTML-Seiten

1.1.3 Hauptleistungen

An das ASV-System werden hinsichtlich der Leistungsmerkmale *Datenumfang* und *Zeit* folgende Anforderungen gestellt:

- Keine Beschränkung der Anzeigenanzahl durch das ASV-System.
  Die Anzahl der Anzeigen, die durch das ASV-System verwaltet werden können, soll nicht durch das Design des ASV-Systems beschränkt sein. Beschränkungen ergeben sich natürlich durch den auf dem Server des Anzeigendienstes zur Verfügung stehenden Speicherplatz.

- Keine Beschränkung der Dateianzahl pro Anzeige durch das ASV-System.
  Jede Anzeige soll theoretisch aus beliebig vielen Dateien bestehen können. Praktisch ist eine Begrenzung der Anzeigengröße durch den Betreiber z.B. auf 1MB angebracht.
  Die durchschnittliche Größe für eine Web-Seite (eine HTML-Seite inklusive GIF-Grafiken oder JPEG-Bilder) sollte nach [NF96] nicht über 30KB liegen, so daß der Platz von 1MB auch für umfangreiche Anzeigen mit mehreren HTML-Seiten ausreichend sein sollte. Dieser Speicherplatz reicht auch aus, um kleine Videos zum Download bereitstellen zu können.

1.1.4 Benutzungsschnittstelle

Die Benutzungsschnittstelle soll sowohl dem Kunden als auch dem Betreiber des ASV-Systems über einen WWW-Browser zur Verfügung stehen und eine ständige Erfolgskontrolle und Korrektur von einmal gemachten Eingaben ermöglichen.

1.1.5 Qualitätsmerkmale

Das System soll stabil gegenüber Fehleingaben sein.

1.2 Prüfen der fachlichen Durchführbarkeit

1.2.1 Problemstellungen

Folgende Problemstellungen ergeben sich aus den im vorangehenden Abschnitt aufgestellten Hauptanforderungen:

1. Problemstellung: Transfer der gestalteten Anzeigen zum Anzeigendienst und Zugriff auf bereits veröffentlichte Anzeigen

Während bei der üblichen Nutzungsweise des WWW die Dateien vom WWW-Browser als Client angefordert und danach vom WWW-Server an den WWW-Browser geliefert und durch ihn dargestellt werden, muß nun eine Möglichkeit gefunden werden, die Dateien der gestalteten Anzeige über den WWW-Browser auszuwählen und zum WWW-Server des Anzeigendienstes zu transferieren. Bei der Anzeigenannahme müssen darüber hinaus auch die Anzeigendaten (Rubrik, Schaltbeginn, Schaltdauer etc.) für die Verwaltung in eindeutiger Form aufgenommen werden.
Da die Kunden nicht über einen (Betriebssystem)-Account auf dem WWW-Server des Anzeigendienstes verfügen, trotzdem aber Veränderungen der Anzeigendateien oder der Anzeigendaten möglich sein sollen, muß die eindeutige Zuordnung der Dateien zu einem Auftrag und des Auftrages zu einem Kunden gewährleistet werden.

2. Problemstellung: Flexibler Zugriff auf die Anzeigendaten, Automatisierung der Auftragsabwicklung

Nachdem eine Anzeige freigeschaltet wurde, soll die Auftragsabwicklung automatisch erfolgen, d.h., daß die Anzeigen ab dem gewünschten Schaltbeginn, für die gewünschte Schaltdauer, in der gewünschten Rubrik usw. angezeigt werden sollen.
Das bedeutet, daß nach den vom Nutzer gewünschten Kriterien jeweils dynamisch eine Übersicht des Anzeigenbestandes aus den Anzeigendaten generiert werden muß.

1.2.2 Techniken für die Realisierung von WWW-Anwendungen

Die Darstellung in diesem Abschnitt erfolgt auf der Grundlage von [LO98].
Zentrales Element jeder WWW-Anwendung ist der *WWW-Server* (siehe Abb. 1). Er nimmt Anfragen des *WWW-Clients* (*Browser*) entgegen (1) und liefert die angeforderten statischen oder dynamisch generierten Inhalte zurück. Dies können HTML-Seiten mit GIF-Grafiken, JPEG-Bildern, MPEG-, AVI- oder Quicktime-Videos, Sound-Dateien sowie Java-Applets sein. Statische Inhalte werden ggf. von einem Proxy geliefert, der häufig angeforderte Dokumente zwischenspeichert, um diese dem Client schneller bereitstellen zu können (2).

Die Kommunikation zwischen Browser und WWW-Server erfolgt dabei mittels des *HyperText Transfer Protokolls* (HTTP).

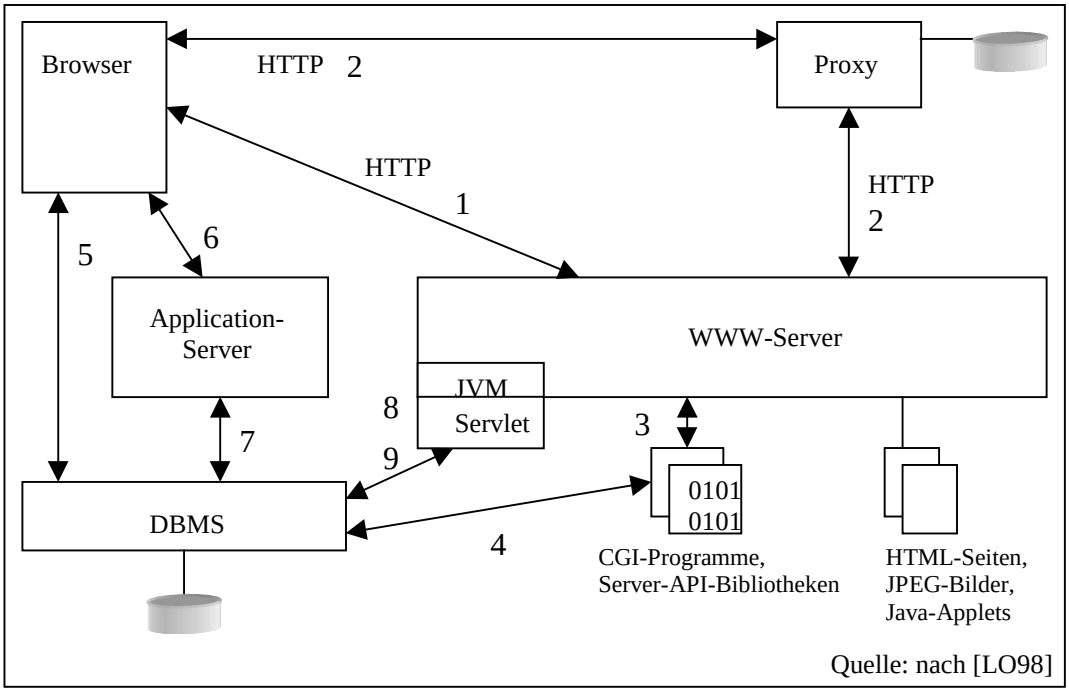

Abb. 1 Techniken für die Realisierung von WWW-Anwendungen

Der WWW-Server übernimmt weiterhin bei speziellen HTTP-Aufrufen das Starten von Anwendungen über das *Common Gateway Interface* (CGI) und von *Server-Erweiterungen* (3) sowie die Ausführung von *Java-Servlets* (8). Werden bei der Ausführung von CGI-Programmen, Server-Erweiterungen oder Servlets HTML-Seiten (nach einem Datenbankzugriff (4,9)) dynamisch generiert, so werden diese vom WWW-Server nach der Abarbeitung des aufgerufenen Programms an den Browser geliefert.

Ein Datenbankzugriff ist aber nicht nur server-seitig über CGI-Programme oder Java-Servlets möglich, sondern kann auch vom Browser aus mittels Java-Applets erfolgen. Dies geschieht entweder direkt (5) oder über einen sogenannten Applikationserver (6,7).

1.2.3 Eine formularbasierte Lösung

Die im vorangehenden Abschnitt dargestellten Techniken für die Realisierung von WWW-Anwendungen kann man in Verfahren, die HTML-Formulare nutzen (CGI-Programme, Server-API, Servlets) und Verfahren, die Java-Applets nutzen (standalone Java-Applets, verteilte Java-Anwendungen), einteilen.

Zum gegenwärtigen Zeitpunkt werden im WWW für interaktive Anwendungen fast ausschließlich formularbasierte Lösungen eingesetzt. Java-Applets kommen dagegen nur sehr selten zum Einsatz, zum Beispiel für das Online-Banking. Mindestens eine Großbank (*Deutsche Bank*) ist inzwischen sogar wieder von der Java-Applet-Lösung abgekommen und bietet Online-Banking mittels HTML-Formularen an.

Daher fiel die Entscheidung, die Durchführbarkeit einer formularbasierten Lösung zu prüfen. Dabei konnte folgende Lösungsmöglichkeit für die aus den Hauptanforderungen abgeleiteten Problemstellungen gefunden werden:

Die Entwicklung und die Protokolle des Internets werden in sogenannten *Requests for Comments* (RFC) dokumentiert. Eine Erweiterung der üblichen Formular-elemente wie Texteingabefelder, Checkboxen und Radiobuttons um eine Auswahlmöglichkeit für Dateien wurde von Nebel und Masinter (*Xerox Corporation)* im RFC 1867[6] vorgeschlagen. Mit dem sogenannten *File-Upload* soll die Beschränkung von herkömmlichen Formularen, die nur Datenabfragen ermöglichen, aufgehoben werden und die Möglichkeit geschaffen werden, den Benutzer zur Übermittlung von Dateien aufzufordern. Der Nutzer kann sich dabei durch einen Klick auf den im RFC vorgeschlagenen *BROWSE*-Button den Inhalt seiner Festplatte anzeigen lassen, eine Datei auswählen und per *SUBMIT*-Button an den Server übertragen, wo sie zur Weiterbearbeitung bereitsteht.

Der WWW-Browser *Netscape Navigator* der Firma *Netscape* unterstützt File-Upload seit Version 2.0 und auch der *Internet Explorer* von *Microsoft* hat das File-Upload-Feature ab Version 4.0 implementiert. Damit würde die Benutzung des File-Upload über Formulare eine Lösung darstellen, die es derzeit einem Großteil aller WWW-Nutzer gestattet, Dateien, nur unter Benutzung eines üblichen Browsers, zu einem WWW-Server zu übermitteln.

Die aktuellen Implementierungen der Browser folgen allerdings nicht in allen Punkten den im RFC 1867 vorgeschlagenen Spezifikationen. So erlauben beide WWW-Browser jeweils nur die Auswahl einer Datei pro Dateiauswahlfeld. Das bedeutet, daß man den Kunden mehrmals zum File-Upload auffordern muß, um eine Anzeige, die zum Beispiel aus mehreren HTML-Dateien besteht, zum Anzeigendienst zu übertragen. Auf diese Weise ist nach jedem File-Upload eine Erfolgskontrolle für den Inserenten möglich, so daß ein File-Upload in mehreren Schritten akzeptabel ist. Die Forderung, die Dateien vor dem Upload in einem Archiv zu bündeln, würde gegen die Anforderung verstoßen, nur einen WWW-Browser zu benötigen, und ist damit nicht akzeptabel.

---

[6] http://www.alternic.com/rfcs/rfc1800/rfc1867.html

Die Nutzung des File-Uploads bietet zudem folgenden Vorteil: Für den Filetransfer im Internet wird auch heute noch üblicherweise das *File Transfer Protokol* (FTP) genutzt. Bei der Nutzung des File-Uploads dagegen wird für die Übertragung der Dateien ebenfalls HTTP genutzt, das bei der Übertragung eine höhere Leistung als FTP erreicht [TG96, S.266].

Um die Dateien später eindeutig den entsprechenden Anzeigendaten zuzuordnen, könnte die für die Anzeige zu erstellende Auftragsnummer auch als Bezeichnung für das Verzeichnis verwendet werden, indem die Dateien gespeichert werden. So ist der Zugriff auf die Dateien für gewünschte Veränderungen über das ASV-System jederzeit möglich.

Bei der Betrachtung der zweiten sich aus den Hauptanforderungen ergebenden Problemstellung (flexibler Zugriff auf die Anzeigendaten, automatisierte Auftragsabwicklung) wird deutlich, daß bei großen Nutzerzahlen ein effizienter Zugriff auf die Anzeigendaten notwendig ist. In [SA94] wird die Speicherung von Daten in Datenbanken vs. Dateisystem diskutiert. Insbesondere die Eigenschaft, daß Anfragen an eine Datenbank mittels einer Anfragesprache wie SQL gestellt werden können und nicht jeweils einzeln programmiert werden müssen, spricht für die Verwendung einer Datenbank. Die Anbindung einer Datenbank an eine WWW-Anwendung ist in [TG96, S.275ff.] sowie in [MA96] ausführlich beschrieben, so daß sich diese Problemstellung im wesentlichen auf den Entwurf eines den Anforderungen gerecht werdenden Datenbankschemas reduziert.

Eine erste Implementierung zeigte, daß sich die gefundene Lösung (Verwendung des File-Uploads nach RFC 1867) praktisch umsetzen läßt. Daraufhin wurde entschieden, die Realisierung des ASV-Systems auf der Grundlage dieser formularbasierten Lösung vorzunehmen.

1.3 Lastenheft

Das Lastenheft ist das fachliche Ergebnisdokument der Planungsphase und enthält eine Zusammenfassung aller fachlichen Basisanforderungen. Die Form ist dabei ein vorgegebenes, standardisiertes, grobes Gliederungsschema mit festgelegten Inhalten [BA96, S.57].

Das ausgearbeitete Lastenheft befindet sich im Anhang A.

2. Definition

Ziel des Definitionsprozesse ist es, aus den Vorgaben der Planungsphase und den von Auftraggeberseite geäußerten unvollständigen, inkonsistenten und oftmals nur vage formulierten Anforderungen ein vollständiges, konsistentes, eindeutiges und durchführbares Anforderungsdokument, die *Produkt-Definition*, zu erstellen [BA96, S.92].
Das Ergebnis der Definitionsphase sind folgende Dokumente [BA96, S. 110]:

1. Pflichtenheft
2. Produktmodell
3. Konzept der Benutzungsoberfläche
4. Benutzerhandbuch

In den folgenden Abschnitten werden die Überlegungen, die zur Erstellung von Pflichtenheft (siehe Anhang B) und Produktmodell (siehe Anhang C) führten, und das Konzept der Benutzungsoberfläche dargestellt. Auf die gesonderte Ausarbeitung eines Benutzerhandbuches wurde verzichtet, da alle wesentlichen Bestandteile wie zum Beispiel eine Übersicht über die zur Verfügung stehenden Funktionen in den folgenden Abschnitten enthalten sind.

2.1 Das Pflichtenheft

Das Pflichtenheft ist das erste fertigzustellende Dokument nach der Planungsphase und enthält eine detaillierte, verbale Beschreibung der fachlichen Anforderungen des zu entwickelnden ASV-Systems. Das Schema ist dabei eine standardisierte Gliederungsform.
Für die Ausarbeitung wurde ein aus dem *IEEE Guide to Software Requirements Specification* (SRS)-Schema abgeleitetes, verkürzte Pflichtenheftschema [BA96 S.104] verwendet. Grundlage für das Pflichtenheft waren dabei das Lastenheft, die Ergebnisse der Durchführbarkeitsuntersuchung sowie folgende Überlegungen:

Mit dem ASV-System werden drei Benutzergruppen arbeiten:

1. Kunden (Inserenten), die Anzeigen aufgeben
2. Verwalter, die Aufträge (neue Anzeigen, Veränderungen) freischalten
3. Nutzer, die Anzeigen lesen

Den drei Benutzergruppen (Kunden, Nutzer, Verwalter) entsprechend wurde die Gruppierung der Funktionen im Pflichtenheft gestaltet: Anzeigenannahme und –service in den Funktionen /F10/-/F110/ und /F120/-/F205/, die Anzeigen-verwaltung in den Funktionen /F210/-/F500W/ und die Anzeigendarstellung in den Funktionen /F510/-/F540/ (vgl. Anhang B). Die Überlegungen zu den einzelnen Funktionsgruppen sind in den Abschnitten 2.1.1 (Anzeigenannahme und –service), 2.1.2 (Anzeigenverwaltung) und 2.1.3 (Anzeigendarstellung), die Überlegungen zur Produkt- und Entwicklungsumgebung im Abschnitt 2.1.4 dargestellt.

2.1.1 Anzeigenannahme und –service

Vor der Annahme der Anzeige sollte sich der Kunde auf einer Login-Seite mittels Zugangsnummer und Passwort beim ASV-System anmelden müssen. Auf diese Weise kann eine eindeutige Zuordnung von Anzeige und Kunde erreicht werden. Damit vermeidet man auch, daß Internet-Nutzer „zum Spaß“ Anzeigen aufgeben und somit die Anzeigenverwaltung erschweren. Erst nach erfolgreicher Anmeldung gelangt er zur eigentlichen Anzeigenannahme.
Da eine Vielzahl von Eingaben vorgesehen sind (Auswahl der zu übermittelnden Dateien, die Anzeigendaten wie Überschrift, Kurzinhalt, Kontakt-Email und die Schaltdaten wie Rubrik, Schaltbeginn und Schaltdauer), ist eine Aufteilung des Anzeigenannahmeprozesses in drei aufeinanderfolgende Schritte sinnvoll:

1. Annahme der Anzeigendateien
2. Annahme der Anzeigendaten
3. Annahme der Schaltdaten

Zum Abschluß sollte dem Kunden eine Übersicht angeboten werden, die noch einmal die wesentlichen Anzeigen- und Schaltdaten enthält. Erst nach einer Bestätigung dieser Daten ist die Anzeige ordnungsgemäß aufgegeben und dem Kunden wird eine Bestätigung angezeigt. Während der Anzeigenannahme muß ein geordneter Abbruch angeboten werden.
Auch beim Anzeigenservice muß sich der Kunde beim ASV-System anmelden. Nach erfolgreicher Anmeldung sollte ihm eine Übersicht mit seinen bisher geschalteten Anzeigen dargeboten werden. So kann er nun für jede Anzeige einzeln auswählen, ob er die Anzeige verändern oder von der Anzeige aussetzen will, d.h., ob die Anzeige für die Internet-Nutzer nicht mehr sichtbar sein soll, etwa, weil der Kunde sein angebotenes Auto schon verkauft hat. Bei kommerziellen Anzeigendiensten ist diese Funktion notwendig, um Kunden die Möglichkeit zu geben, Produkte, die ausverkauft sind, nicht weiter zu bewerben. Das Aussetzen sollte allerdings noch nicht dem Löschen der Anzeige gleichkommen, sondern eine Vorstufe darstellen. Das Löschen sollte erst nach dem regulären Ablauf der Schaltdauer erfolgen, um dem Kunden die Möglichkeit zu geben, eine Anzeige auch nur kurzzeitig auszusetzen, ohne sie danach wieder komplett neu aufgeben zu müssen.
Möchte der Kunde seine Anzeige verändern, so sollte er den gleichen Prozeß wie bei der Anzeigenannahme durchlaufen, mit dem Unterschied, daß ihm bereits gespeicherte Daten angezeigt und zur Veränderung angeboten werden. Es sollte ihm insbesondere möglich sein, Anzeigendateien hinzuzufügen, zu löschen oder durch eine neue Version zu ersetzen.
Ebenfalls sinnvoll wäre die Möglichkeit, die Schaltdauer einer Anzeige verlängern zu können, ohne die gesamte Anzeige mit Anzeigendateien, Anzeigen- und Schaltdaten noch einmal neu aufgeben zu müssen.

2.1.2 Die Anzeigenverwaltung

Die Anzeigenverwaltung dient dem Freischalten eingegangener Aufträge (Neuaufträge, Veränderungs- bzw. Verlängerungsaufträge).
Der Verwalter sollte nach der Anmeldung mit seiner Zugangsnummer und seinem Passwort auf der Login-Seite eine Statistik über die Anzahl der neu eingegangenen Anzeigen und der vorliegenden Veränderungs- und Verlängerungsaufträge

erhalten. Wählt sich der Verwalter zum Beispiel über einen Provider ins Internet und danach in das ASV-System ein und sieht er, daß keine neuen Anzeigen vorliegen, kann er sich gleich wieder aus dem ASV-System ausloggen und die Verbindung ins Internet trennen, was Kosten spart.
Auf der Statistikseite sollte eine Auswahlmöglichkeit für die Funktionen

- Neue Anzeigen freischalten,
- Veränderungen freischalten und
- Verlängerungen freischalten (Wunschkriterium)

existieren. Nach Auswahl einer dieser Funktionen sollte der Verwalter auf eine Übersichtsseite gelangen, wo ihm je nach Auswahl alle neu eingegangenen, alle veränderten Anzeigen bzw. alle Verlängerungsaufträge angezeigt werden.
Zu jedem Auftrag sollten dem Verwalter folgende Bearbeitungsmöglichkeiten angeboten werden:

- Anzeige/Veränderung/Verlängerung freischalten (sofortiges Freischalten ohne weitere Bearbeitung)
- Anzeige/Veränderung/Verlängerung bearbeiten (Detailansicht der Anzeigen- und Schaltdaten mit Möglichkeit für Veränderungen und Möglichkeit der Voransicht der Anzeige)
- Anzeige/Veränderung/Verlängerung verwerfen (Ablehnen eines Auftrages)

Auf diese Weise hat der Verwalter die Möglichkeit, Aufträge bei Bedarf sehr schnell abzuarbeiten. Er kann andererseits aber auch eine genaue Kontrolle jeder neu eingegangenen Anzeige oder Anzeigenveränderung vor dem Freischalten vornehmen.

Bei Anwahl der Funktion *Anzeige/Veränderung/Verlängerung bearbeiten* sollte der Verwalter zu einer Detailansicht der Anzeige gelangen und hier die Möglichkeit haben, sich eine Voransicht des Inserates anzeigen zu lassen sowie letzte Änderungen vor dem Freischalten vorzunehmen. Kann sich der Verwalter vorläufig nicht entscheiden, den Auftrag freizuschalten, oder hat er versehentlich Veränderungen vorgenommen, die er rückgängig machen möchte, sollte ihm die Möglichkeit angeboten werden, den Auftrag zurückzustellen, wobei er wieder zur Auftragsübersicht gelangt.
Wie auch bei Anzeigenannahme und -service muß ein geordneter Abbruch (Ausloggen aus dem Verwaltungssystem) jederzeit möglich sein.

2.1.3 Die Anzeigendarstellung

Der Nutzer des ASV-Systems sollte zwischen mehreren Vorgehensweisen wählen können. Erstens, er weiß genau, was er sucht, und möchte die vorhandenen Anzeigen möglichst schnell nach bestimmten Schlagwörtern durchsuchen. Dafür ist eine Eingabemöglichkeit für Schlagwörter einzurichten. Das Ergebnis der Suche sollte dem Nutzer in Form einer Übersicht mit den Anzeigenüberschriften und der Kurzbeschreibung dargestellt werden. Er kann nun eine Anzeige auswählen und betrachten.
Die zweite Vorgehensweise ist das „Surfen im Anzeigenbestand“. Der Nutzer sucht nicht nach einer speziellen Anzeige, sondern möchte sich einen

umfassenden Überblick über die vorhandenen Anzeigen verschaffen. Dafür sollte ihm die Möglichkeit angeboten werden, durch den Anzeigenbestand zu navigieren.

2.1.4 Produkt- und Entwicklungsumgebung

In Kapitel 1.2.3 wurde die Verwendung einer formularbasierten Lösung für die Realisierung des ASV-Systems diskutiert.
Als Entwicklungswerkzeuge wurden daraufhin die Programmiersprache *PHP* und die Datenbank *MySQL* gewählt. PHP und MySQL werden als Teil des *LAMP*-Systems in [KU98] vorgestellt. Die Buchstaben stehen dabei für *L*inux (Betriebssystem), *A*pache (WWW-Server), *My*SQL (relationale Datenbank) und *P*HP (Programmiersprache). Während Betriebssystem und WWW-Server in der Regel durch den Betreiber des WWW-Servers vorgegeben werden und daher nicht näher betrachtet werden sollen, ist die Verwendung von PHP und MySQL für die Realisierung des ASV-Systems zu begründen.

Für die Verwendung von PHP und MySQL sprechen folgende Argumente:

- PHP, eine Programmiersprache mit C-ähnlichem Syntax, bietet alle Funktionen, die für Realisierung des ASV-Systems benötigt werden: Unterstützung von File-Upload nach RFC 1867, Erstellung von HTTP-Headern und mächtige Datenbankfunktionen für viele kommerziell oder frei verfügbare Datenbanken wie die Datenbanken von *Sybase* und *Oracle* sowie *Postgres*, *mSQL*, u.a.
- PHP ist sowohl für Unix-Derivate wie *AIX* von *IBM*, *HP-UX* von *HP*, *IRIX* von *SGI* oder *Linux* als auch für *WindowsNT* erhältlich und ermöglicht so eine plattformübergreifende Entwicklung.
- MySQL ist eine relationale Datenbank, die den *SQL92 Entry Level* implementiert und ebenfalls für Unix und WindowsNT erhältlich ist.
- MySQL für Unix und PHP für Unix/NT sind frei erhältlich und ohne Lizenzgebühren sowohl privat als auch kommerziell einsetzbar. Eine Ausnahme macht nur MySQL für WindowsNT. Hier wird eine geringe Lizenzgebühr fällig.
- MySQL und PHP können ohne Beteiligung des *Root*-Nutzers (unter Unix) installiert werden und belegen nur wenige MB Plattenplatz – wichtig bei der Anmietung von Plattenplatz auf einem Server.

Gegen die Verwendung von PHP und MySQL sprechen folgende Punkte:

- PHP-Code wird als Quellcode in HTML-Seiten eingebettet und dann durch das eigentliche PHP-Programm, den *Hypertext Preprocessor*, verarbeitet. Damit wird PHP also interpretiert, ein Geschwindigkeitsnachteil gegenüber der Verwendung einer kompilierbaren Programmiersprache wie zum Beispiel *C*.
- Der Quellcode ist dadurch bei einer Weitergabe des ASV-Systems an Dritte nicht geschützt.
- MySQL fehlt ein Transaktionsmechanismus.

PHP kann aber als Apache-Modul kompiliert werden, womit der Nachteil normaler CGI-Anwendungen umgangen werden kann, daß bei jedem Aufruf ein neuer Prozess für die Anwendung gestartet werden muß. Dafür ist allerdings die Unterstützung des Root-Nutzers (unter Unix) notwendig. Da aber das ASV-System nur als Grundlage für die Entwicklung von Anzeigen-Online-Diensten dienen soll, ist eine Weitergabe des Quellcodes von vornherein vorgesehen. Zudem wurde die Entwicklung eines PHP-Compilers bereits angekündigt. Beim Einsatz des ASV-Systems ist der Quellcode für Kunden, Nutzer und Verwalter dagegen nicht sichtbar.
Einzeloperationen werden von MySQL atomar ausgeführt. Sollte es notwendig werden, für eine Reihe von Einzeloperationen einen Transaktionsmechanismus nachzubilden, so können behelfsmäßig die MySQL-eigenen Lock-Mechanismen verwendet werden.

Da die Nachteile von PHP und MySQL nicht gegen eine Verwendung sprechen, wurde aufgrund der genannten Vorteile die Entscheidung zugunsten von PHP und MySQL getroffen.

Das ausgearbeitete Pflichtenheft befindet sich in Anhang B.

2.2 Das Produktmodell

Zusätzlich zum Pflichtenheft muß eine fachliche Problemlösung stattfinden, indem die Daten und Funktionen des ASV-System mittels formaler Konzepte modelliert werden. Dafür wurde die Methode der *Strukturierten Analyse* (SA) nach [BA96, S.397ff] mit den Basiskonzepten *Datenflußdiagramm* (DFD), *Data Dictionary* (DD) und *Mini-Spezifikationen* (MiniSpec) sowie zusätzlich das Basiskonzept *Entity-Relationship-Modell* (ER-Modell) verwendet. Die Erarbeitung des ER-Modells, des DFD, des DD und der MiniSpec erfolgte dabei unter Berücksichtigung der Anforderungen des Pflichtenheftes und durch die Analyse der Problemstellungen, die sich aus der Realisierung des ASV-Systems als verteilte *Client/Server*-Anwendung ergeben.

Die Ergebnisdokumente ER-Modell, DFD, DD und MiniSpec (in Form von *Pseudocode*) befinden sich im Anhang C.

2.2.1 Das ASV-System als Client/Server-Anwendung

Durch die Anforderung, daß verschiedene, geografisch verteilte Personen mittels eines WWW-Browsers als Client über das WWW zur gleichen Zeit Anzeigen auf einem zentralen Server aufgeben, ändern oder verwalten sollen, ergeben sich Problemstellungen hinsichtlich der Überprüfung der Authentizität eines Kunden oder Verwalters oder der Wahrung der Integrität der Daten. Bei den nachfolgenden Betrachtungen stehen prinzipielle Lösungsmöglichkeiten im Vordergrund. Zum Beispiel: Welche Daten müssen gespeichert werden, um eine Authentifikation durchführen zu können?

Problemstellung: Überprüfung der Authentizität eines Kunden oder Verwalters

Für den Betreiber des ASV-Systems ist es entscheidend sicherzustellen, daß ein Kunde oder Verwalter, der sich beim ASV-System anmelden möchte, auch wirklich die Person ist, für die er sich ausgibt. Und dies sowohl bei der erstmaligen Beantragung des Zuganges zum ASV-System als auch bei jedem Einloggen in das ASV-System über das WWW. Dies ist zum einen notwendig, um eine unberechtigte Nutzung des bereitgestellten Zugangs zu verhindern, zum anderen, um Ansprüche gegenüber Kunden (z.B. Rechtmäßigkeit der Inhalte, bei kommerziellen Diensten die Zahlung von Schaltgebühren) durchsetzen zu können. Aber auch intern kann es wichtig sein, festzustellen, welcher Mitarbeiter z.B. für eine Freischaltung verantwortlich war.

Problemlösung

Bevor ein Kunde Zutritt zum ASV-System erhält, muß er einen Zugang (*Login*) beim Betreiber des ASV-Systems beantragen. Hat ein Kunde einen Zugang zum ASV-System über das Internet beantragt, kann der Betreiber zuerst eine Überprüfung der Authentizität der übermittelten Daten mittels Telefon, Fax oder in ganz sensiblen Fällen durch Überprüfung des Personalausweises (oder einer Kopie davon) vornehmen. Ist sich der Betreiber sicher, daß die Person wirklich existiert und die Daten korrekt sind, kann er in einer Datenbank die Daten des

Kunden sowie eine Kundennummer und ein Kundenpasswort eintragen. Dem Kunden werden nun Kundennummer und Passwort mitgeteilt. Ebenso werden für einen Verwalter eine Verwalternummer und ein Verwalterpasswort sowie die Daten des Verwalters in die Datenbank eingetragen.

Will sich der Kunde oder Verwalter einloggen, muß er nun nur noch Kunden- bzw. Verwalternummer und Passwort angeben. Die Authentifikation eines Kunden oder Verwalters reduziert sich damit auf die Überprüfung des zu einer Kunden- oder Verwalternummer gehörenden Passwortes.

Daten

Kunde:= Kundennummer+Kundenpasswort+Name
Verwalter:= Verwalternummer+Verwalterpasswort+Name

Problemstellung: Wahrung der Integrität – Verhinderung der Erstellung mehrerer Aufträge gleichen Typs für dieselbe Anzeige

Einem Kunden soll es ermöglicht werden, zur gleichen Zeit mehrmals im ASV-System eingeloggt zu sein, um z.B. während des Schaltens einer neuen Anzeige Informationen über bereits geschaltete Anzeigen zu erhalten. Dabei muß jedoch verhindert werden, daß er für eine Anzeige, für die er einen Auftrag eines bestimmten Typs erstellt oder erstellt hat, weitere Aufträge gleichen Typs, d.h. weitere Veränderungs- oder weitere Verlängerungsaufträge, erstellt, bevor der zuerst erteilte Veränderungs- bzw. Verlängerungsauftrag freigeschaltet wurde. Würde man dies nicht verhindern, könnte man Mißverständnisse nicht ausschließen: Bei nur zwei Veränderungsaufträgen gleichzeitig für eine Anzeige hätte man bereits zu entscheiden, ob nun nur der erste, nur der zweite oder beide Veränderungsaufträge hintereinander ausgeführt werden sollen.

Problemlösung

Mit der Entscheidung, einen Auftrag zu erstellen, wird ein Eintrag dieses Auftrages in der Datenbank angelegt, wobei die Anzeige, die durch diesen Auftrag betroffen ist, und der Auftragstyp verzeichnet sind. Damit ist diese Anzeige für diesen Auftragstyp gesperrt.
Beim Versuch, einen weiteren Auftrag gleichen Typs zu dieser Anzeige anzulegen, passiert nun folgendes:

- Wenn der erste Auftrag bereits abgeschlossen ist, wird die Annahme eines weiteren Auftrages dieses Typs für diese Anzeige verweigert.
- Wenn der erste Auftrag noch in Bearbeitung ist, erscheint eine Wahlmöglichkeit, den ersten Auftrag zu löschen und mit dem zweiten fortzufahren oder die Bearbeitung des zweiten abzubrechen.

Daten

Auftrag:= betrifftAnzeige+Auftragstyp+abgeschlossen

Problemstellung: Ergänzung I - Löschen eines nicht ordnungsgemäß abgeschlossenen Auftrages durch den Kunden

Ein Kunde hat einen Auftrag zu einer Anzeige angelegt, die Bearbeitung aber z.B. wegen eines Systemabsturzes nicht ordnungsgemäß abschließen können. Es existiert nun ein Auftragseintrag zu dieser Anzeige und die Anzeige ist somit gesperrt – zumindest für einen Auftrag gleichen Typs.

Problemlösung

Dem Kunden, der einen neuen Auftrag gleichen Typs erstellen möchte, wird angeboten, den Auftrag ordnungsgemäß abzubrechen (siehe oben), woraufhin der Auftragseintrag in der Datenbank gelöscht wird. Nun kann ein neuer Auftrag gleichen Typs geschaltet werden.

Problemstellung: Ergänzung II - Erkennen und Löschen eines nicht ordnungsgemäß abgeschlossenen Auftrages eines Kunden durch die Verwaltung

Wird ein Auftrag zu einer Anzeige nicht ordnungsgemäß abgeschlossen (Bedienungsfehler, Systemabsturz), so existiert er dennoch, da er bereits angelegt wurde. Der Kunde könnte nun dennoch diese Anzeige bearbeiten, indem er von der Möglichkeit Gebrauch macht, den ersten Auftrag zu löschen (siehe oben). Löscht der Kunde den abgebrochenen Auftrag jedoch nicht, bleibt ein nicht abgeschlossener Auftrag bestehen. Für die Verwaltung ist nun nicht nachvollziehbar, ob der Auftrag gerade noch in Bearbeitung ist und deshalb noch nicht abgeschlossen, oder ob er wirklich nicht ordnungsgemäß abgeschlossen wurde.

Problemlösung

Bei jedem Einloggen in die Verwaltung wird festgestellt, ob der Kunde, der einen noch nicht abgeschlossenen Auftrag geschaltet hat, noch eingeloggt ist oder nicht. Wenn nein, wird dieser Auftrag gelöscht. Das bedeutet, daß das Einloggen und Ausloggen eines Kunden in Form einer *Sitzung* protokolliert werden muß. Bei jeder Aktion wird nun auch die Zeit dieser Aktion vermerkt. So kann der Kunde automatisch ausgeloggt werden, falls er z.B. länger als eine halbe Stunde lang keine Aktion mehr ausgeführt hat. Somit kann eindeutig festgestellt werden, wann ein nicht ordnungsgemäß abgeschlossener Auftrag gelöscht werden kann.

Daten

Sitzung:= Kundennummer+Zeit letzter Aktion

Problemstellung: Wahrung der Integrität - Verhinderung der gleichzeitigen Bearbeitung eines Auftrages durch mehrere Verwalter

Im Gegensatz zu Kunden soll nun nicht nur ein Verwalter mehrmals zur gleichen Zeit eingeloggt sein können, sondern es sollen auch mehrere Verwalter zur gleichen Zeit Zugriff auf den gleichen Auftragsbestand haben. Bearbeitet nun ein Verwalter einen Auftrag, so muß sichergestellt werden, daß andere Verwalter diesen nicht bearbeiten dürfen – der Auftrag muß für sie gesperrt sein. Hat nun der Verwalter, der eine Anzeige bearbeitet, einen Systemabsturz oder bricht er die Bearbeitung nicht ordnungsgemäß ab, so daß die Sperre nicht wieder aufgehoben

werden kann, so ist der Auftrag weiterhin gesperrt für andere – und für den Verwalter selbst.
Weiterhin muß verhindert werden, daß ein Verwalter, der einen Auftrag bereits bearbeitet, diesen ein weiteres Mal zur Bearbeitung aufrufen kann, da dies zu Inkonsistenzen führen kann. (Man denke nur an eine Textverarbeitung, die es ermöglicht, verschiedene Stände eines Dokuments gleichzeitig geöffnet zu haben. Die Korrekturen in dem einen Dokument vernichten bei der Speicherung alle anderen Korrekturen.)

Problemlösung

Entscheidet sich ein Verwalter, einen Auftrag zu bearbeiten, so wird seine Verwalternummer in den Auftrag eingetragen, womit dieser gesperrt ist. Wie auch beim Kunden muß entscheidbar sein, wann ein Verwalter nicht mehr eingeloggt ist, um eventuell von ihm vorgenommene Sperrungen zu löschen. Daher wird auch für Verwalter der Einlogg- und Ausloggvorgang in einer Sitzung protokolliert.

Es können nun folgende Fälle vorkommen (siehe Abb. 2), wenn derselbe Verwalter (V=1) oder ein anderer Verwalter (V=0) versucht, Zugriff auf einen gesperrten Auftrag zu erhalten, wobei die Auftragsübersicht die Sperrung anzeigt (Ü=1) oder nicht anzeigt (Ü=0) und der Verwalter, der den Auftrag gesperrt hat, noch eingeloggt (e=1) oder nicht mehr eingeloggt (e=0) ist und der Auftrag inzwischen abgeschlossen (a=1) oder (noch) nicht ordnungsgemäß abgeschlossen (a=0) wurde. Es sind somit 2^4 = 16 Fälle zu beachten. Ist bei einem erneuten Bearbeitungsversuch der Auftrag bereits abgeschlossen, muß die Bearbeitung abgewiesen werden. Es verbleiben somit 16 – 8 +1 = 9 Fälle. Wenn der Verwalter, der den Auftrag gesperrt hat, nicht mehr eingeloggt ist, kann er auch keinen Zugriff mehr versuchen. Die Sperrung ist diesem Verwalter automatisch zu entziehen. Es verbleiben somit 9 – 2 = 7 zu beachtende Fälle. Die 7 interessanten Fälle sind in Abb. 2 aufgeführt (V=Verwalter, Ü=Übersicht, e=eingeloggt, a=abgeschlossen).

Daten

Sitzung:= Sitzung+Verwalternummer+Zeit der letzten Aktion
Auftrag:= Auftrag+Verwalternummer

2.2.2 Das Datenflußdiagramm

Das DFD beschreibt die Wege von Daten bzw. Informationen zwischen Funktionen, Speichern (z.B. Datenbank oder Dateien) und die Transformation der Daten bzw. Informationen durch Funktionen. Die Vorstellung dabei ist, daß das System bereits läuft, und man überprüft, ob mit den Daten, die in eine Funktion hineinfließen, die Ausgangsdaten generiert werden können.
Die Vorteile eines DFD sind seine leichte Erstellbarkeit und die gute Vermittelbarkeit an Dritte (z.B. den Auftraggeber). Nachteilig wirkt sich bei der Reinform eines DFD aus, daß es schnell unübersichtlich wird, weshalb auf die hierarchischen DFD der Strukturierten Analyse zurückgegriffen wurde. Ausgehend von einem abstrakten Datenflußdiagramm (*Kontextdiagramm*) wird jeder Prozeß verfeinert und durch ein eigenes Datenflußdiagramm beschrieben.

V	Ü	e	a	Beschreibung	Reaktion
?	?	?	1	Die Bearbeitung des Auftrages wurde abgeschlossen, er ist nicht mehr existent.	Der Bearbeitungsversuch wird abgewiesen. Die Übersicht wird aktualisiert.
0	0	0	0	Ein anderer Verwalter möchte diesen Auftrag bearbeiten, wobei seine Übersicht die Sperrung nicht anzeigt, da er sich vor der Sperrung eingeloggt hat. Der Verwalter, der den Auftrag gesperrt hat, ist nicht mehr eingeloggt und hat die Auftragsbearbeitung nicht ordnungsgemäß abschlossen, so daß die Sperrung nicht aufgehoben wurde.	Der andere Verwalter erhält die Sperrung und kann den Auftrag bearbeiten.
0	1	0	0	Wie 0000, nur Übersicht zeigt Sperrung an.	Dem Verwalter wird keine Bearbeitungsmöglichkeit angeboten und ein Bearbeitungsversuch abgewiesen. Beim nächsten Einloggen werden aber dann alle Aufträge, die gesperrt und nicht ordnungsgemäß abschlossen wurden und deren Verwalter nicht mehr eingeloggt sind, automatisch gelöscht und die Übersicht aktualisiert.
0	0	1	0	Wie 0000, nur ist der Verwalter, der den Auftrag gesperrt hat, noch eingeloggt.	Der Bearbeitungsversuch des anderen Verwalters wird abgewiesen, seine Übersicht aktualisiert.
0	1	1	0	Ein anderer Verwalter möchte diesen Auftrag bearbeiten, wobei seine Übersicht die Sperrung anzeigt, da er sich nach der Sperrung eingeloggt hat. Der Verwalter, der den Auftrag gesperrt hat, ist noch eingeloggt und hat die Bearbeitung noch nicht abgeschlossen.	Eine Bearbeitung des Auftrages wird nicht angeboten. Der Name und die Verwalternummer des sperrenden Verwalters werden angegeben.
1	0	1	0	Der gleiche Verwalter möchte diesen Auftrag bearbeiten, wobei seine Übersicht die Sperrung nicht anzeigt, da er sich vor der Sperrung ein weiteres Mal eingeloggt hat. Die Bearbeitung des Auftrages ist noch nicht abgeschlossen.	Der Verwalter erhält die Mitteilung, daß er diesen Auftrag schon bearbeitet und alle gemachten Veränderungen verfallen, falls er auf einer erneuten Bearbeitung besteht. Besteht er auf der Übernahme der Bearbeitung, erhält er in einer neuen Sitzung den alten Stand und kann die Bearbeitung nun ordnungsgemäß abschließen. Die alte Sitzung wird entwertet. Er erhält aber auch die Möglichkeit, den Auftrag freizugeben. Falls er im alten Fenster nun die Bearbeitung fortsetzen möchte, scheitert dies an der entwerteten alten Sitzung.
1	1	1	0	Wie 1010, wobei seine Übersicht die Sperrung anzeigt, da er sich nach der Sperrung ein weiteres Mal eingeloggt hat.	Dem Verwalter wird keine Bearbeitungsmöglichkeit für diese Anzeige geboten.

Abb. 2 Zugriffskonflikte in der Verwaltung

Ist der Prozeß ausreichend verstanden, erfolgt keine weitere Verfeinerung. Statt dessen erfolgt eine nähere Beschreibung durch eine Mini-Spezifikation, die sogenannte MiniSpec.
Beim Übergang vom Kontextdiagramm, das die Schnittstellen des Systems zur Umwelt modelliert, zum DFD 0 (oberste Hierarchieebene) wurde deutlich, daß an dieser Stelle die Erarbeitung der notwendigen *Speicher* für das Datenfluß-diagramm durch ein ER-Modell notwendig wurde. Die Entwicklung der DFD wurde daher unterbrochen und ein ER-Modell erarbeitet, das die Grundlage für die ersten Entwürfe der DFD bildete. Im weiteren Erarbeitungsprozess wurden die Verfeinerung der DFD und die Anpassung des ER-Modells parallel durchgeführt.

2.2.3 Das Entity-Relationship-Modell

Das Ergebnis der Modellierung der Sicht auf die Daten ist das ER-Modell. Das ER-Modell stellt Entity-Typen (abstrakte oder physische Objekte der realen Welt mit gemeinsamen Eigenschaften) und Relationship-Typen (Zusammenfassung von gleichartigen Beziehungen zwischen Entities, die jeweils gleichen Entity-Typen angehören), sowie die Darstellung von Attributen und die Vereinbarung von Primärschlüsseln für die Modellierung bereit [RA97, S. 3-5ff].

Folgende fachliche Problemstellungen standen bei der Modellierung im Mittelpunkt:

1. Modellierung der Objekte *Kunde* und *Verwalter*
2. Modellierung der Objekte *Anzeige* und *Auftrag*

2.2.3.1 Modellierung der Objekte Kunde *und* Verwalter

Für die Modellierung standen drei Möglichkeiten zur Verfügung. Erstens könnten *Kunde* und *Verwalter* getrennt als zwei eigenständige Entity-Typen aufgefaßt werden, die später auch getrennt in Tabellen *Kunde* und *Verwalter* der Datenbank gespeichert werden (siehe Abb. 3a). Im Programm würde dann anhand einer mitgeführten *BenutzerID* festgestellt werden können, welche Tabelle der Datenbank bei Abfragen der Zugangsnummer und des Benutzerpaßwortes abzufragen ist.
Zweitens könnte ein Entity-Typ *Person* Ausgangspunkt der Modellierung sein, der Attribute speichert, die sowohl *Kunde* als auch *Verwalter* beschreiben, z.B. *Zugangsnummer*, *Paßwort*, *Name* (siehe Abb. 3b). Die beiden Entity-Typen *Kunde* und *Verwalter* könnten danach mittels *Spezialisierung* aus *Person* gebildet werden. *Kunde* und *Verwalter* stünden dann in einer *is-a*-Beziehung zu *Person*. Bei der Abbildung des ER-Modells auf das relationale Datenmodell wird die Abbildung einer *is-a*-Beziehung jedoch nicht direkt unterstützt und führt, falls sie dennoch durchgeführt wird, zu einem zusätzlichen Suchaufwand [RA97, S.4-18]. Dies spricht gegen diese zweite Möglichkeit.
Drittens könnte man sich nur für einen Entity-Typ *Benutzer* entscheiden und die Unterscheidung in *Kunde* und *Verwalter* mittels eines zusätzlichen Attributes, zum Beispiel *Benutzerart*, explizit machen (siehe Abb. 2c). Diese Modellierung könnte wie die erste Möglichkeit ebenfalls direkt auf das relationale Datenmodell abgebildet werden. Bei einer Erweiterung der Attributmenge um sowohl für Kunden als auch Verwalter gültige Attribute wie *Anschrift*, *Bankverbindung*, usw. würde dieser dritte Ansatz die Vorteile der zweiten Möglichkeit (gemeinsame

Attribute nur einmal vereinbaren) nutzen, ohne einen zusätzlichen Suchaufwand zu verursachen.

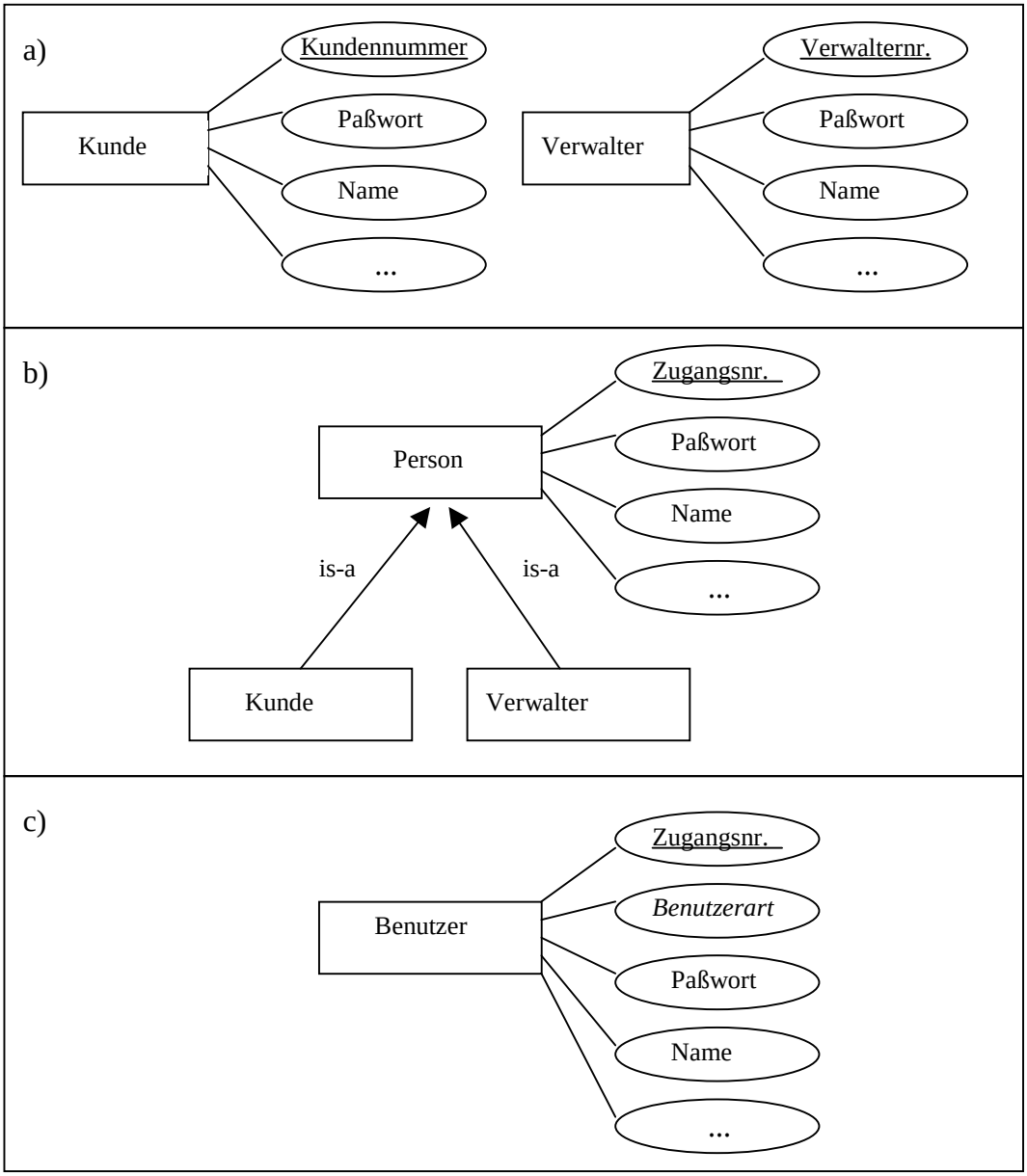

Abb. 3 Modellierung von *Kunde* und *Verwalter*

Die erste Möglichkeit (getrennte Entity-Typen *Kunde* und *Verwalter*) wäre demnach nur von Vorteil, wenn eine kleine Anzahl von gemeinsamen Attributen und eine größere Anzahl von unterschiedlichen Attributen zu erwarten ist. Dies ist aber für das ASV-System nicht anzunehmen, so daß die Entscheidung zugunsten der dritten Möglichkeit gefällt wurde.

2.2.3.2 Modellierung der Objekte Anzeige *und* Auftrag

Während Kunden und Verwalter in der ASV-Miniwelt nicht in einer Beziehung stehen, die eine Modellierung mittels eines Relationship-Typs sinnvoll gemacht hätte, bietet sich bei der Modellierung von *Anzeige* und *Auftrag* die Einführung eines Relationship-Typs an, schließlich wird ein Auftrag zu einer Anzeige geschaltet.

Ohne die Einführung dieser Relationship könnte man diese Zugehörigkeit im Modell nicht darstellen, was gegen die Einführung von zwei getrennt aufgeführten Entity-Typen *Anzeige* und *Auftrag* spricht (siehe Abb. 4a).

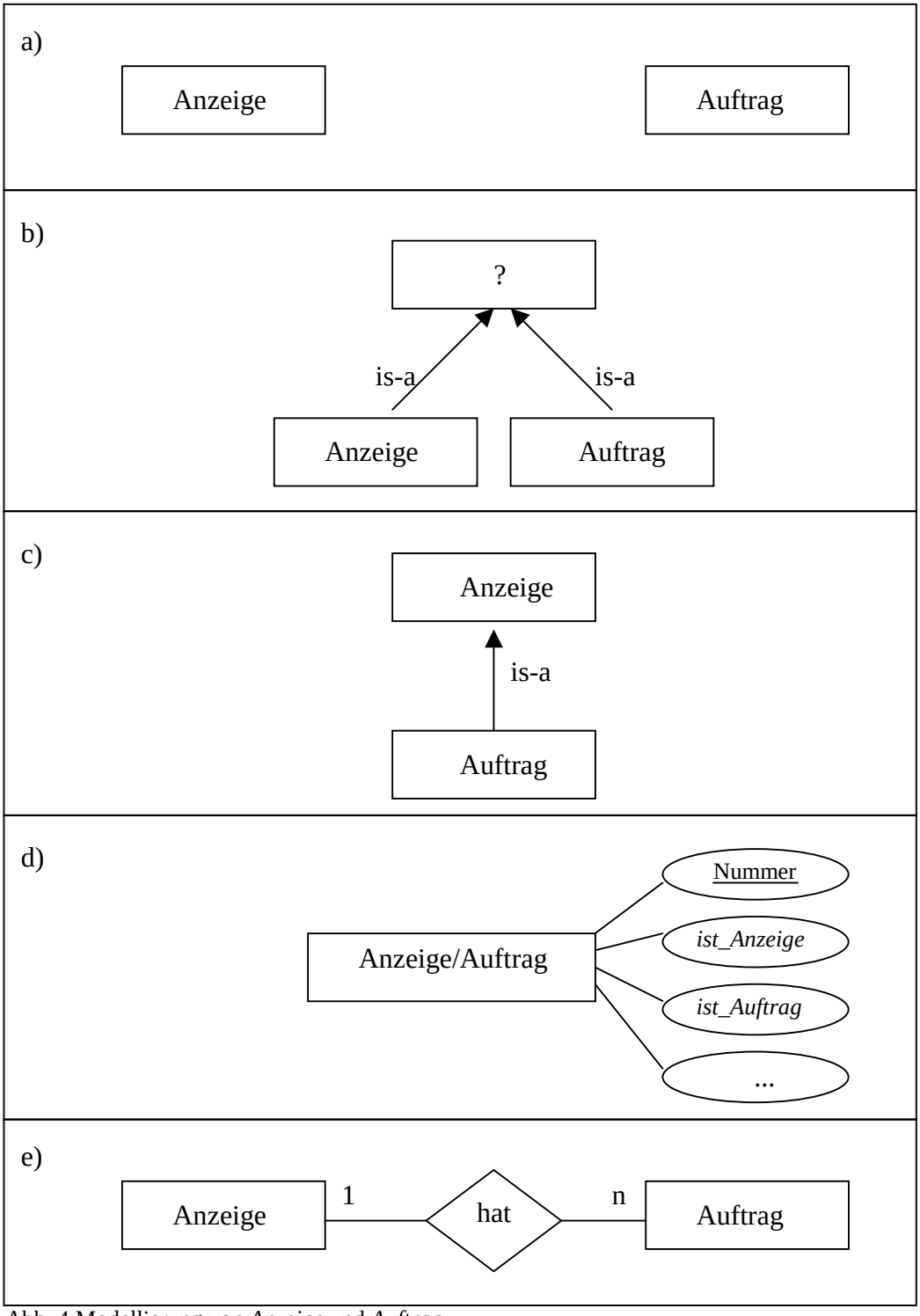

Abb. 4 Modellierung von *Anzeige* und *Auftrag*

Weiterhin hätte man auch hier die Möglichkeit, einen Entity-Typ einzuführen, von dem ausgehend *Anzeige* und *Auftrag* mittels Spezialisierung gebildet werden könnten (siehe Abb. 4b). Schließlich ergibt sich ohne diesen übergeordneten Entity-Typ eine strukturelle Redundanz: Sowohl *Anzeige* als auch *Auftrag* hätten dann jeweils die Attribute *Anzeigenüberschrift*, *Start-HTML-Seite*, *Schlagwörter*,

Kontakt-Email-Adresse, usw. Gegen diese Lösung spricht, daß sie sich erstens nicht direkt auf das relationale Datenmodell abbilden läßt und zweitens die Semantik der ASV-Miniwelt nicht realitätsnah abbildet: Eine *Anzeige* und ein *Auftrag* sind zwei verschiedene Objekte unserer Vorstellung, die sich nur „künstlich“ von einem übergeordneten Objekt ableiten lassen, wobei die Ableitung von *Kunde* und *Verwalter* von dem übergeordneten Objekt *Benutzer* ganz „natürlich“ möglich ist.

Es verbleiben somit noch zwei Fälle: Man könnte die Auffassung vertreten, daß gilt: *Auftrag is-a Anzeige* (siehe Abb. 4c). Dies trifft jedoch bei Verlängerungsaufträgen nicht zu. Ein *Auftrag* erbt zudem nicht alle Attribute von *Anzeige*, womit die *is-a*-Beziehung nicht erfüllt wird. Zum Beispiel soll die Entität *Anzeige* über die Attribute *ausgesetzt* und *abgelaufen* verfügen, welches aber für einen *Auftrag* keine Bedeutung besitzen. Bei der Wahl einer *is-a*-Beziehung zwischen *Auftrag* und *Anzeige* würde sich auch folgendes Problem ergeben:
Bei der Erteilung eines Veränderungsauftrages, wobei der Kunde sowohl die Anzeigendaten als auch Teile der Schaltdaten ändern können soll (Pflichtenheftfunktionen /F140-200/), würde in der aus dem Entity-Typ *Auftrag* gebildeten Tabelle zwar ein Veränderungsauftrag gespeichert, die alten Anzeigendaten jedoch in der aus dem Entity-Typ *Anzeige* gebildeten Tabelle überschrieben. Die Folge wäre, daß die Anzeige mit Erteilung eines Auftrages ausgesetzt werden müßte, da ja der Verwalter erst den Inhalt der veränderten Anzeige überprüfen können soll. Würde der Verwalter nun entscheiden, daß er den Veränderungsauftrag nicht annimmt, müßte die Anzeige noch einmal komplett neu aufgegeben werden, was jeden Kunden verärgern würde. Die Lösung hierfür wäre die Speicherung der veränderten Anzeigendaten unter einer neuen Anzeigennummer und die gleichzeitige Aufnahme von alter und neuer Anzeigennummer in einem *Auftrag* – eine nicht sehr elegante Lösung.

Auch die Einrichtung eines einzigen Entity-Typ ähnlich wie der Entity-Typ *Benutzer*, wobei ein Attribut wie zum Beipiel *ist_Anzeige* oder *ist_Auftrag* die Unterscheidung ermöglicht (siehe Abb. 4d), wurde aufgrund der größeren Anzahl unterschiedlicher Attribute wie *Neuauftrag*, *Verlängerungsauftrag*, *Auftragsdatum*, *abgeschlossen*, usw. nicht gewählt.

Damit fiel die Entscheidung für die Lösung, zwei Entity-Typen *Anzeige* und *Auftrag* einzuführen, wobei die Zuordnung eines Auftrag zu einer Anzeige mittels einer Relationship modelliert wird (siehe Abb. 4e).

Das erarbeitete ER-Modell für das ASV-System befindet sich in Anhang C. Die bei der Deklaration eines Entity-Typs notwendige Festlegung der Wertebereiche der Attribute wurde dabei erst in der Entwurfsphase (siehe Abschnitt 3.1) durchgeführt.

2.3 Das Konzept der Benutzungsoberfläche

Allen drei Benutzergruppen (Kunden, Nutzer, Verwalter) ist gemeinsam, daß sie die Anzeigen in Form von HTML-Seiten mit Links und eingebundenen Grafiken sehen. Die Benutzerführung sollte dem Rechnung tragen, indem die Oberflächengestaltung durch HTML-Gestaltungselemente und die Eingabe von Daten über HTML-Formularseiten erfolgt. So findet kein Gestaltungsbruch zwischen der Oberfläche des ASV-Systems und den eigentlichen Anzeigen statt. Dies entspricht auch der Forderung nach einer Benutzungsschnittstelle, die nur mit einem üblichen HTML-Browser angezeigt werden soll (siehe Abschnitt I.3.1).
Bei einer HTML-Formular-basierten Benutzerführung stehen für die Gestaltung der Benutzungsoberfläche folgende Gestaltungselemente zur Verfügung:

- HTML-Gestaltungselemente: formatierter Text, Tabellen, Links etc.
- HTML-Formulargestaltungselemente: Eingabezeile, Passworteingabezeile, Auswahlbox, Mehrfachauswahlbox, Radio-Buttons, Check-Boxen etc.

Die Gestaltung von Dialogen mittels HTML-Formularen hat dabei einige Besonderheiten. Die Überprüfung der Eingabedaten kann bei HTML-Formularen erst dann erfolgen, wenn das Formular wieder an den Server zurückgeschickt wird. Dieser generiert nun entweder den Folgediaglog oder liefert das gleiche Formular zusammen mit einer Fehlermeldung an den Benutzer zurück. Das kostet jedoch immer Übertragungszeit. Die einfachste Möglichkeit, die Anzahl eventueller Rücksendungen zu minimieren, besteht darin, ausführliche Erläuterungstexte in die Dialoge einzubauen. Da es sich bei HTML-Formularen um normale HTML-Seiten handelt, können Dialoge entwickelt werden, die über mehrere Bildschirmseiten gehen und vom Benutzer im Browserfenster gescrollt werden. Eine Alternative besteht in der Verwendung von *Java-Script*, einer Eigenentwicklung der Firma Netscape, die auch von Browsern von Microsoft interpretiert werden kann. Java-Script-Befehle werden in die HTML-Seite integriert und nach dem Laden im Browserfenster ausgeführt. Es handelt sich also hierbei um eine Form der client-seitigen Programmausführung. So können zum Beispiel Eingabefenster schon vor dem Rücksenden des Formulars an den Server auf korrekte Eingaben überprüft werden oder dynamisch die Vorbelegung von Eingabefenstern verändert werden.
Die Verwendung von Java-Script ist eine Option. Die Nutzung von Java-Script muß aber im Einzelfall entschieden werden: Zum einen vergrößert die Verwendung von Java-Script die Ladezeit der Formularseiten, da das komplette Java-Script mitgeladen werden muß. Zum anderen kann der Benutzer ohne weiteres die Ausführung von Java-Script durch das Deaktivieren von Java-Script in seinem Browser unterbinden. Wie die Erfahrung zeigt, deaktivieren Nutzer oft die Ausführung aktiver Inhalte wie Java oder Java-Script, um die Manipulation ihres Rechners aufgrund möglicher Sicherheitslücken auszuschließen. Somit müßte man trotzdem wieder eine server-seitige Überprüfung durchführen, da Nutzer auf eine Meldung wie „*Zum Benutzen dieses Services müssen Sie Java-Script aktivieren*“ durchaus mit der „Nicht-Nutzung“ reagieren können.

Für das ASV-System sollten daher HTML-Formularseiten ohne Java-Script zum Einsatz kommen.

Bei der Analyse des Pflichtenheftes wurden folgende Funktionskomplexe identifiziert, die für einen reibungslosen Annahme- und Verwaltungsprozeß besonders wichtig sind. Sie werden anschließend in den Abschnitten 2.3.1 bis 2.3.4 untersucht:

- Gestaltung der Serviceseite für einen Kunden mit Bestandsübersicht und Funktionsauswahl (Pflichtenheft /F130/)
- Gestaltung des Upload-Dialoges für die Kunden (Pflichtenheft /F20-70/)
- Benutzerführung in der Anzeigenverwaltung (Pflichtenheft /F210-F500W/)
- Gestaltung der Anzeigenübersichten und –suche für die Nutzer (Pflichtenheft /F510-F540/)

2.3.1 Gestaltung der Serviceseite für einen Kunden

Im Pflichtenheft sind die Funktionskomplexe Anzeigenannahme (/F20-F110/) und Anzeigenservice (/F120-F205/) getrennt dargestellt. Da hierbei der Kunde dem Betreiber des Anzeigen-Online-Dienstes in jedem Fall einen Auftrag erteilt, sollte die Benutzerführung beide Funktionskomplexe vereinen. D.h., daß sich der Kunde nicht getrennt für die Anzeigenannahme und den Anzeigenservice einloggen muß. Nachdem er sich mittels Kundennummer und Passwort in das ASV-System eingeloggt hat, gelangt der Kunde demnach zu seiner Serviceseite, die ihm

- eine Bestandsübersicht der von ihm geschalteten Anzeigen,
- das Schalten von neuen Anzeigen sowie die Funktionen *Verändern*, *Verlängern*, *Aussetzen/Einsetzen* je Anzeige und
- die Möglichkeit, sich geordnet auszuloggen (Exit-Möglichkeit),

anbietet. Ein Bestandseintrag soll dabei aus der Anzeigennummer, der Anzeigenüberschrift sowie der Restlaufzeit je Anzeige bestehen. Hierbei kommen zwei unterschiedliche Gestaltungsmöglichkeiten in Frage: Einmal die Auflistung in Form einer Tabelle (siehe Abb. 5) oder aber ein gesonderter Einzeleintrag in Listenform je Anzeige (siehe Abb. 6). Für die Tabelle spricht, daß sie durch die uniforme Darstellung eine schnelle Übersicht über die Daten aller Anzeigen ermöglicht.

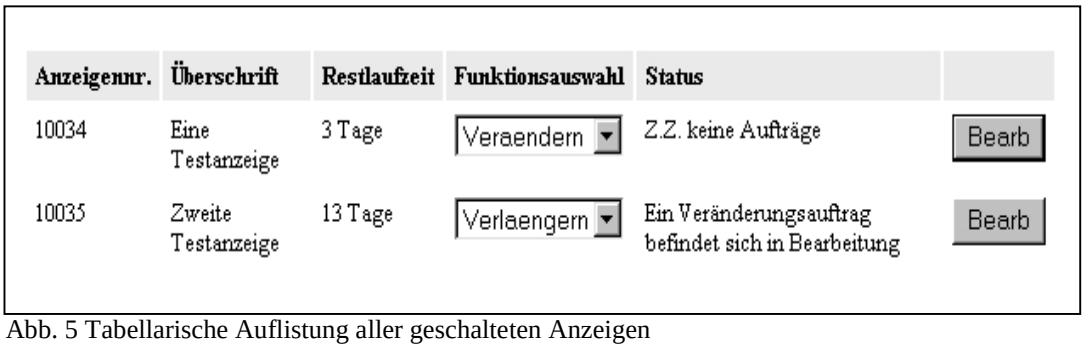

Abb. 5 Tabellarische Auflistung aller geschalteten Anzeigen

Sie ist somit für eine große Anzahl von geschalteten Anzeigen die geeignetere Darstellungform. Denkbar wäre bei einer Tabelle auch das Angebot unterschiedlicher Sortierungskritierien wie die Sortierung nach Auftragsnummer (äquivalent: nach Auftragsdatum), nach Rubrik oder nach Restlaufzeit.

Zu beachten ist, daß die Funktionsauswahl für jede Anzeige einzeln angezeigt werden sollte. Dies ist notwendig, damit der Kunde weiß, welche Auftragsart er für diesen Auftrag schalten kann. Die Alternative einer allgemeinen Funktionsauswahl würde im Einzelfall (ein Verlängerungsauftrag wurde zum Beispiel geschaltet, aber von der Verwaltung noch nicht freigeschaltet) zur einer Ablehnungsmitteilung führen, was aber erst nach Absenden des Formulars an den Web-Server ermittelt werden kann. Die generierte Seite mit der Ablehnungsmitteilung muß danach wieder vom Browser des Kunden geladen werden, was Frustrationsgefühle beim Kunden erzeugen kann. In der Tabellendarstellung würde das bedeuten, daß zusätzlich in einer Spalte die Funktionsauswahl dargestellt wird, entweder in Form von Radio Buttons oder als Auswahlbox, sowie eine Statusspalte, die den Bearbeitungsstatus anzeigt, wenn bereits Aufträge vorliegen (siehe Abb. 5).

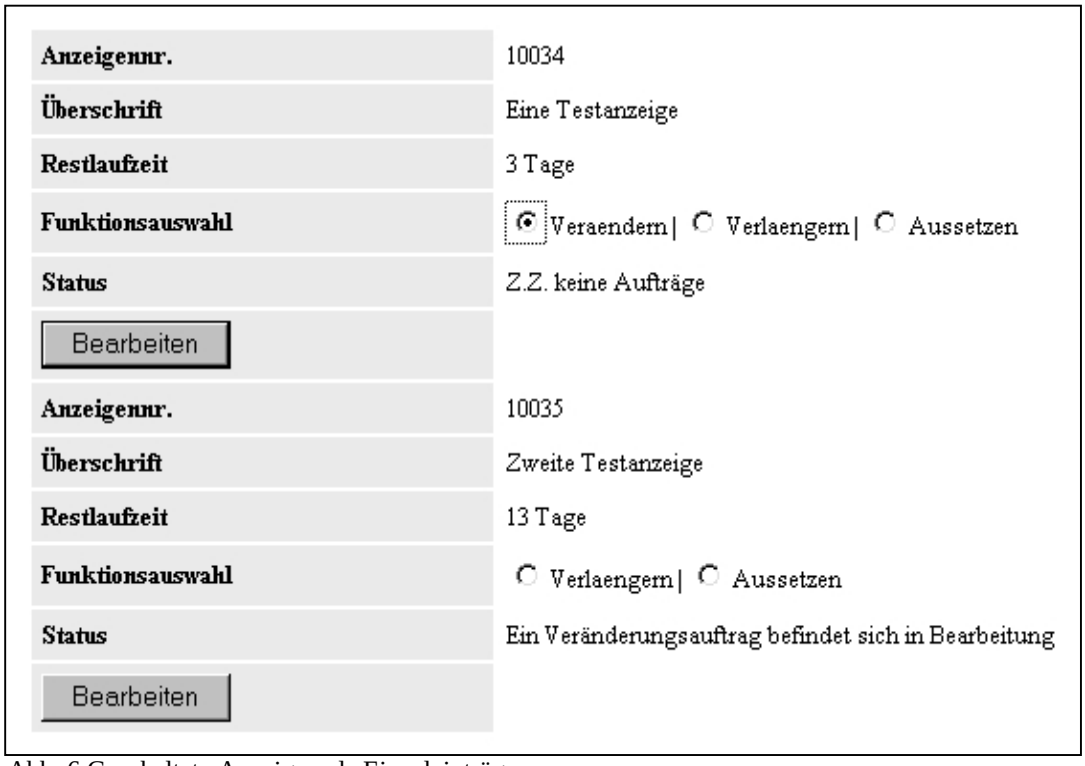

Abb. 6 Geschaltete Anzeigen als Einzeleinträge

Die Realisierung von Einzeleinträgen dagegen ist eher für eine kleinere Anzahl von Anzeigen geeignet, da die Übersichtlichkeit nicht wie in einer Tabelle gegeben ist. Sie ermöglicht aber eine Darstellung, die es dem Kunden ermöglicht, sich besser auf die einzelne Anzeige zu konzentrieren, da der Einzeleintrag alle Daten der Anzeige optisch gebundener darstellt (siehe Abb. 6).

Für die Realisierung des ASV-Systems sollte eine Mischform gewählt werden, wobei die Auftragsnummer in einer Extra-Spalte, abgesetzt von dem Listeneintrag, dargestellt wird. So ist ein schnelles Auffinden einer Anzeige über die Auftragsnummer möglich, gleichzeitig wird nach dem Auffinden die Erfassung aller notwendigen Angaben (Restlaufzeit, zur Verfügung stehende Funktionen) durch den geschlossenen Eintrag erleichtert (siehe Abb.7).

2.3.2 Gestaltung des Upload-Dialoges für die Kunden

Nach der Entscheidung für das Schalten einer neuen Anzeige oder eines Veränderungsauftrages gelangt der Kunde zum Upload-Dialog, der ihm

- das Hochladen von ausgewählten Dateien und die Übersicht über bereits hochgeladene Dateien bietet,
- das Löschen von bereits hochgeladenen Dateien und
- die Voransicht seiner Anzeige ermöglicht, wobei er die Startseite frei wählen können soll.

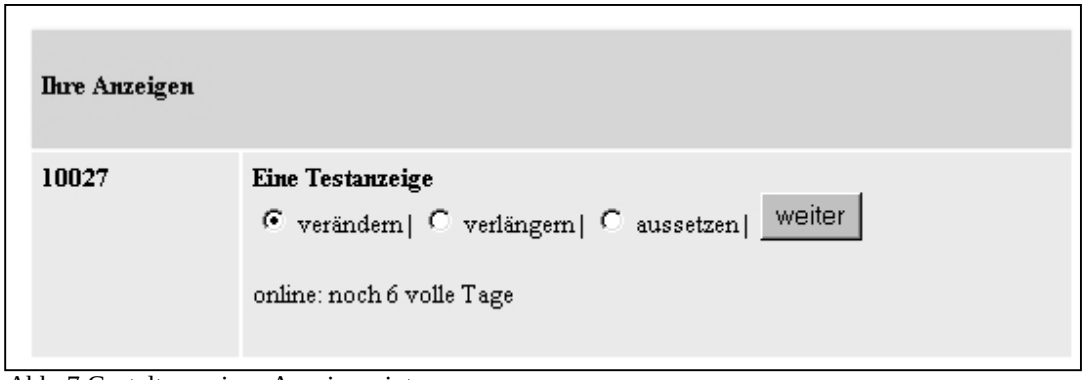

Abb. 7 Gestaltung eines Anzeigeneintrages

Eine erste Entscheidung ist hierbei die Zusammenfassung dieser Funktionen auf einer Formularseite oder die Verteilung über mehrere Formularseiten. Hochladen und Löschen könnten dabei eine Formularseite bilden, die Auswahl der Startseite mit Voransicht eine zweite. Da diese Funktionen aber eine Einheit bilden, sollten sie dem Kunde auch auf einer Formularseite angeboten werden. Dabei erscheint es sinnvoll, dem Kunde die Funktionen *Datei löschen*, *Startseite auswählen* und *Voransicht* erst dann anzubieten, wenn er bereits eine Datei hochgeladen hat.

Wie bereits in Abschnitt 1.2.3 erwähnt, ermöglicht die in den Browsern aktuell implementierte Version der RFC 1867 nur die Auswahl jeweils einer hochzuladenden Datei je Upload-Vorgang. Das bedeutet, daß der Kunde nach jeder Datei, die er ausgewählt hat, den Upload-Button betätigen und den Upload abwarten muß. Dieses Verfahren hat den klaren Nachteil, daß es zeitaufwendig ist. Andererseits bekommt der Kunde so nach jedem erfolgreichen Upload eine Bestätigung. Außerdem erreicht das System eine kürzere Interaktionszeit, da die Zeit für einen einzelnen Datei-Upload mit nachfolgender Ausgabe der Bestätigung natürlich kürzer ist, als wenn mehrere Dateien hochgeladen würden. Wie bereits mehrfach erwähnt, kann nämlich bei der Verwendung einer CGI-basierten Lösung erst dann eine Bestätigung generiert werden, wenn der komplette Vorgang abgeschlossen ist.
Um das gleichzeitige Hochladen mehrerer Dateien in einem Bearbeitungsschritt zu ermöglichen, könnte man dem Kunden mehrere Dateiauswahldialoge auf einer Seite anbieten. Dies würde den Kunden aber möglicherweise irritieren: Der Kunde möchte vier Dateien hochladen, die Oberfläche bietet aber zum Beispiel nur drei Auswahlfelder (siehe Abb. 8). Man müßte nun erklären, daß er die vierte Datei im zweiten Durchgang hochladen muß.

Auch hier wäre vorstellbar, erst die Anzahl der hochzuladenden Dateien abzufragen und dann die benötigte Anzahl von Feldern zur Verfügung zu stellen, was aber auch nicht vollkommen befriedigt.

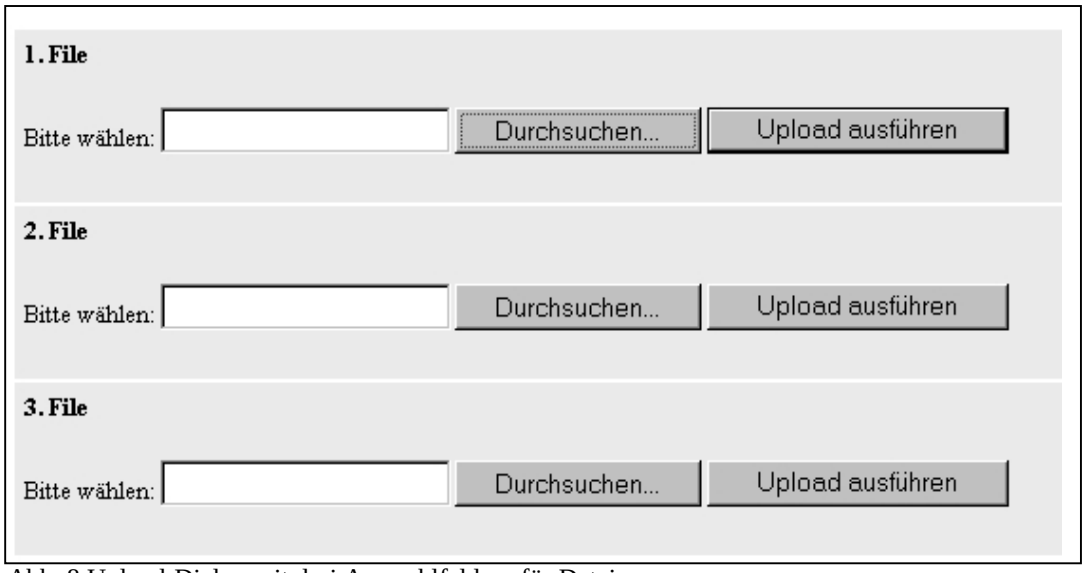

Abb. 8 Upload-Dialog mit drei Auswahlfeldern für Dateien

Für die Realisierung des Upload-Dialoges sollte deshalb das wiederholte Hochladen jeweils einer Datei gewählt werden (siehe Abb. 9).

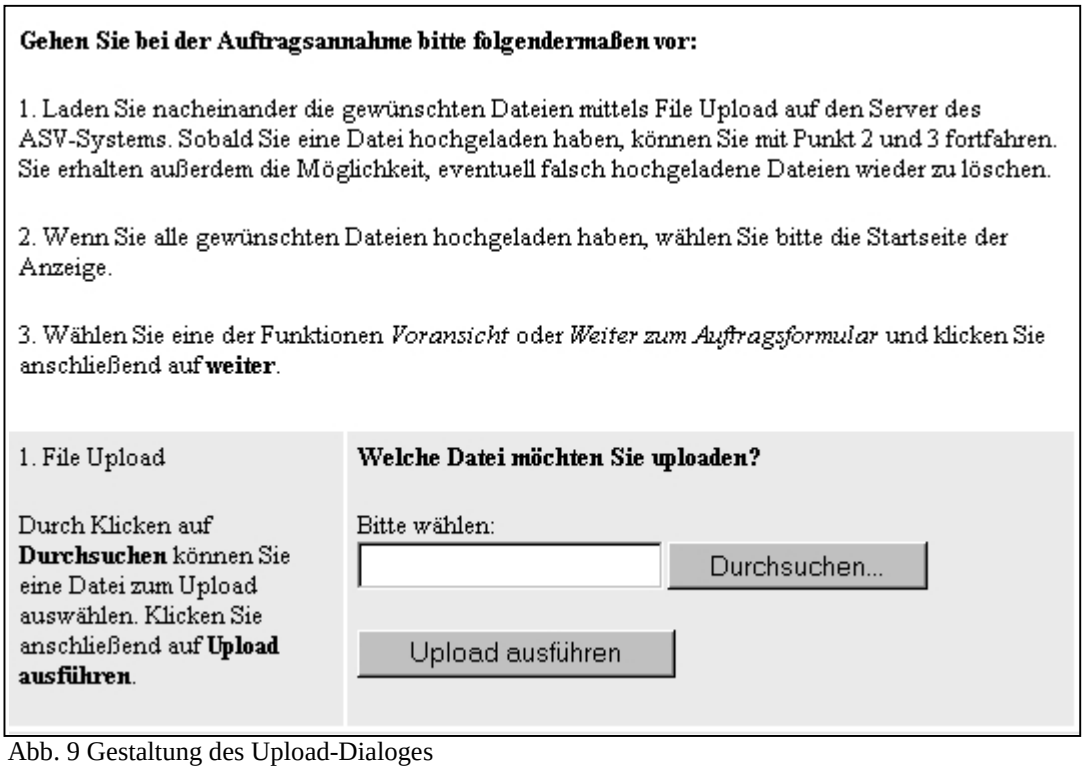

Abb. 9 Gestaltung des Upload-Dialoges

2.3.3 Benutzerführung in der Anzeigenverwaltung

Bei der Gestaltung der Benutzungsoberfläche für die Anzeigenverwaltung sollten folgende Anforderungen berücksichtigt werden:

- Eingegangene Aufträge müssen vom Verwalter schnell abgearbeitet werden können.
- Bei Bedarf müssen auch eine ausführliche Begutachtung des eingegangenen Auftrages und Änderungen möglich sein.

Besonders Verlängerungsaufträge eignen sich für ein sofortiges Freischalten ohne vorherige Begutachtung, da keine Veränderungen an Anzeigendateien oder -daten vorgenommen wurden. Somit können auch keine „verbotenen Inhalte“ ohne Wissen des Verwalters abgelegt werden. Dagegen bedürfen Neu- oder Veränderungsaufträge einer Voransicht, um die Rechtmäßigkeit der angebotenen Inhalte überprüfen zu können. Aber auch hier sollte der Verwalter ein sofortiges Freischalten wählen können, wenn es sich zum Beispiel um einen vertrauenswürdigen Kunden handelt. Bevor jedoch der Verwalter die Entscheidung treffen kann, ob er eine Anzeige sofort freischaltet oder erst einer Betrachtung unterzieht, müssen ihm jeweils alle eingegangenen Aufträge der gewählten Auftragsart (Pflichtenheftfunktion /F230/) angezeigt werden.

Auch hier stellt sich die Frage nach der Darstellungsform. Die Vor- und Nachteile von tabellarischer Darstellungsweise vs. Einzeleinträgen wurden bereits in Abschnitt 2.3.1 diskutiert. Da in der Verwaltung mit einer größeren Anzahl von abzuarbeitenden Anzeigen gerechnet werden muß, sollte diesmal die Entscheidung für die tabellarische Darstellungsform fallen.
Anzuzeigen sind – je Auftrag - neben Kundenname und Auftragsüberschrift eine Check-Box und ein *Bearbeiten*-Button. Eine Funktionsauswahl über und unter der tabellarischen Darstellung der eingegangenen Aufträge mit den Einträgen *markierte Aufträge sofort freischalten* und *markierte Aufträge verwerfen* ermöglicht die schnelle Abarbeitung. Für den Fall, daß ein Auftrag bereits durch einen anderen Verwalter bearbeitet wird, soll anstelle der Check Box und des *Bearbeiten*-Buttons der Name des Bearbeiters erscheinen.
Ein Verwalter, der zum Beispiel alle Verlängerungsaufträge sofort freischalten möchte, kann nun folgendermaßen vorgehen: Er läßt sich die eingegangenen Verlängerungsaufträge anzeigen, markiert alle Verlängerungsaufträge mittels der Check Boxen und wählt *markierte Aufträge sofort freischalten*.

Eine weitere Problemstellung ist die Gestaltung der Benutzerführung und Benutzungsoberfläche für den Fall, daß der Verwalter den Auftrag vor dem Freischalten bearbeiten möchte. Dafür bieten sich drei Lösungsmöglichkeiten an:

- die Verwendung des von der Anzeigenannahme her bekannten Dialoges,
- ein vollkommen neu gestalteter Bearbeitungsdialog,
- eine modifizierte Version des bekannten Annahmedialoges.

Die Verwendung des von der Anzeigenannahme her bekannten Dialoges hätte den Vorteil, daß man dem Verwalter „automatisch“ die gleiche Funktionalität bereitstellt, über die auch der Kunde bei der Auftragsannahme verfügen kann.

Somit sind Änderungen an Dateien sowie an allen Anzeigen- und Schaltdaten zum Bearbeitungszeitpunkt möglich. Ein weiterer Vorteil ist, daß der Inhaber des Anzeigen-Online-Dienstes fortlaufend ein Feedback über die Qualität des Annahmedialoges erhalten wird – schließlich sind die Verwalter darauf angewiesen, ihrer Tätigkeit möglichst effizient nachgehen zu können. Ein möglicher Nachteil ist die vorgesehene Dreitteilung in Upload-Dialog, Annahme der Anzeigendaten und Annahme der Schaltdaten. Sie bedeutet, daß der Verwalter zwischen den Formularseiten navigieren muß, um alle Daten zu sehen.
Das könnte für eine vollkommene Neugestaltung des Bearbeitungsdialoges sprechen. Diese Lösung hätte zudem den Vorteil, daß man sie unabhängig vom Annahmedialog an die Anforderungen der Verwalter anpassen könnte. Dies wird aber gleichzeitig wieder zu einem Nachteil, wenn nun bei jeder Änderung im Annahmedialog die gleiche Änderung ein zweites Mal im neu gestalteten Verwalterdialog implementiert werden muß.

Für die Realisierung des Bearbeitungsdialoges im ASV-System sollte deshalb eine modifizierte Version des von der Anzeigenannahme her bekannten Dialoges gewählt werden. Hierbei sieht der Verwalter die ursprünglich drei Formularseiten zusammengefaßt auf einer Formularseite und kann somit alle Daten überblicken und auf einmal ändern. Bei der Implementierung könnte man dafür direkt auf die vorhandenen Formularseiten des Annahmedialoges zurückgreifen.

2.3.4 Gestaltung der Anzeigenübersichten und –suche für die Nutzer

Laut Pflichtenheft (/F510-540/) soll der Nutzer die Möglichkeit haben, sich den Anzeigenbestand entweder strukturiert nach den Kriterien Rubriken und Kundenname präsentieren zu lassen oder aber direkt nach bestimmten Schlagwörtern zu suchen. Die Problemstellung lautet demnach, eine leicht zu bedienende, optisch ansprechende Navigationsleiste zu entwerfen.

Dafür wurden sieben der erfolgreichsten Web-Kataloge und Suchmaschinen (*yahoo.com, dino-online.de, web.de, lycos.com, alta-vista.com, excite.com* und *fireball.de*) auf folgende Fragestellungen hin untersucht:

- Wie erfolgt die strukturierte Darstellung der Einträge und die Navigation? (Zugriff über Auswahlmenüs oder schrittweises „Hinabsteigen“ in der Hierarchie über Links)
- Falls sowohl eine Schlagwortsuche als auch ein strukturierter Zugriff angeboten werden: Wie ist die Verbindung zwischen Schlagwortsuche und strukturierter Darstellung gelöst? (Getrennte Seiten für Schlagwortsuche und strukturierte Darstellung oder Integration in eine Navigationsleiste)
- Wie wurde die Eingabemöglichkeit für Schlagwörter gestaltet? (Anordnung auf gleicher Seite wie das Ergebnis der Anfrage oder in seperatem Frame)

Dabei konnte eine sehr große Übereinstimmung zwischen den unterschiedlichen Diensten festgestellt werden.

Alle Dienste bieten sowohl eine direkte Schlagwortsuche als auch eine strukturierte Darstellung. Die Eingabemöglichkeit für die Schlagwortsuche befindet sich dabei bei den meisten Diensten direkt auf der Startseite und wird

auch auf den Ergebnisseiten der Suche mitgeführt. Eine Ausnahme bildet hier nur der Web-Katalog dino-online.de, der für die Suche nach Schlagwörtern eine separate Seite eingerichtet hat, die man über einen Link erreicht. Allen Diensten ist weiterhin gemeinsam, daß sie die Eingabemöglichkeit für Schlagwörter nicht in einem separaten Frame darstellen.

Bei der Darstellung der Ergebnisse der Schlagwortsuche differieren die einzelnen Angebote: yahoo.com, lycos.com, alta-vista.com, excite.com und fireball.de zeigen die Ergebnisse nur nach Relevanz geordnet an. Dagegen bieten dino-online.de und web.de eine nach Rubriken und Unterrubriken geordnete Ergebnisdarstellung. Dies verwundert bei den klassischen Suchmaschinen wie lycos.com und alta-vista.com nicht, wurden doch die strukturierten Darstellungen bei diesen Suchmaschinen erst nachträglich erarbeitet. Sie sind so wahrscheinlich nicht mit dem Bestand der ursprünglichen Datenbank verbunden. Bei yahoo.com dagegen hätte eine strukturierte Präsentation der Ergebnisse möglich sein müssen, da yahoo.com als Web-Katalog entwickelt wurde.
Bis auf dino-online.de und excite.com ist die Eingabemöglichkeit für die Schlagwortsuche immer auch bei der Erschließung der Einträge über die Rubrikenhierarchie präsent.

Bei der strukturierten Darstellung herrscht ebenfalls weitgehende Übereinstimmung der Lösungen. Auf der Startseite werden die Rubriken meist tabellarisch dargestellt. Mittels Links gelangt man zur Auflistung der Unterrubriken, bis man schließlich zu den tatsächlichen Einträgen vordringt. Eine Ausnahme hinsichtlich der grafischen Darstellung macht hierbei nur der Dienst alta-vista.com, der die Rubrikenhierarchie in Form von geschachtelten Menüs darstellt. Eine Navigation über Auswahlmenüs bietet dagegen nur einer der untersuchten Dienste: excite.com. Befindet man sich hier in einer der Unterrubriken, kann man per Auswahlmenü direkt in eine andere Rubrik (oberste Hierarchieebene) springen.
Die großen Übereinstimmungen der Gestaltung der Benutzerführung lassen darauf schließen, daß es sich um bewährte Lösungsansätze handelt. Für die Realisierung des Nutzer-Interfaces des ASV-Systems sollte deshalb folgende Lösung zum Einsatz kommen:

Die Schlagwortsuche ist auf jeder Seite präsent. Die Ergebnisse der Schlagwortsuche werden nach Rubriken geordnet angezeigt. In den Bereich der Eingabemöglichkeit für die Schlagwörter ist außerdem eine Auswahlmöglichkeit für die strukturierte Darstellung integriert. Sie ermöglicht die Auswahl, ob die strukturierte Darstellung nach Rubriken oder nach Inserentennamen (Kundennamen) erfolgen soll. Bei Auswahl nach Rubriken werden die Rubriken tabellarisch in Form von Links angezeigt. Bei Auswahl einer Rubrik gelangt der Nutzer dann zu einer – wiederum tabellarisch dargestellten – Übersicht der Unterrubriken. Alle Anzeigen, für die keine Unterrubrik festgelegt wurde, werden hier schon mit Anzeigenüberschrift und Kurzbeschreibung aufgeführt.

3. Entwurf

Aufgabe des Entwurfes nach [BA96, S. 632] ist es, aus den Anforderungen, die in der Definitionsphase festgelegt wurden, eine *Softwarearchitektur* zu entwickeln. Eine Softwarearchitektur ist dabei die Struktur des Softwaresystems, dargestellt durch *Systemkomponenten* (Funktionen/Prozeduren, Klassen) und ihre Beziehungen untereinander. Die Methode des *Strukturierten Designs* (SD) ermöglicht es hierbei, eine Softwarearchitektur zu entwickeln, die aus hierarchisch angeordneten *funktionalen Modulen* (Realisierungen von Funktionen) besteht.

Die Methode SD entstand ursprünglich vor der Methode SA (siehe Abschnitt 2.2), die dann aus dem SD entwickelt wurde. So ging man anfangs beim SD davon aus, daß die Definitionsphase noch nicht durchgeführt wurde. Durch die historische Entwicklung fand dann die Beachtung von Faktoren, die die Softwarearchitektur beeinflussen, auch Eingang in die SA und somit in die Definitionsphase. Zu diesen Faktoren zählen:

- Einsatzbedingungen (z.B. Einbenutzer-/Mehrbenutzer-Betrieb, sequentieller/ nicht-sequentieller Produkteinsatz),
- Umgebungs- und Randbedingungen (z.B. Zielplattform, Datenbanken, Systemsoftware) und
- Qualitätsanforderungen (z.B. Zuverlässigkeit, Änderbarkeit).

Im Fall des ASV-Systems stellte sich heraus, daß wesentliche Aspekte des Entwurfs wie zum Beispiel die Grundsatzentscheidung hinsichtlich der Datenhaltung und der Benutzungsoberfläche, die Beachtung der obengenannten Faktoren und die Festlegung einer Struktur des Systems durch Funktionen in Planungs- und Definitionsphase bereits sehr ausführlich ausgearbeitet wurden. Durch die Anwendung des SD waren daher keine wesentlichen neuen Erkenntnisse zu erwarten.

Somit stand ein Aspekt der Entwurfsphase, der Datenbankentwurf, im Vordergrund. Er wird im Abschnitt 3.1 ausführlich behandelt. Einen weiteren sehr wichtigen Aspekt stellt die Sicherheit dar. Einzelne Teilaspekte wie die Benutzer-Authentifikation wurden auf konzeptioneller Ebene bereits in Abschnitt 2.2.1 besprochen. In Abschnitt 3.2 werden Problemstellungen behandelt, die die Sicherheit der Anwendung betreffen, sowie Lösungsmöglichkeiten angegeben.

3.1 Datenbankentwurf

In der Definitionsphase fiel die Entscheidung hinsichtlich der Datenhaltung zugunsten einer relationalen Datenbank. Der Datenbankentwurf dient nun dazu, aus dem *konzeptionellen Schema* der Datenbank (ER-Modell des ASV-Systems, siehe Abschnitt 2.2.3) durch eine *relationale Datenmodellierung* zuerst zu einem *logischen Schema* und durch Anwendung der sogenannten *Normalisierung* sowie der Festlegung von *Views* und *Zugriffsrechten* schließlich zum *Datenbankschema* zu gelangen, welches die Grundlage für die Implementierung darstellt. Die einzelnen Schritte wie relationale Datenmodellierung und Normalisierung sind im folgenden beispielhaft für den Entity-Typ *Auftrag* dargestellt.

3.1.1 Relationale Datenmodellierung

Mittels relationaler Datenmodellierung entsteht aus dem konzeptionellen Schema (ER-Modell) das logische Schema der Datenbank. Dabei sind folgende Teilaufgaben auszuführen [BA96, S. 714]:

- Festlegen der Relationenschemata mit Namen, Attributen und deren Wertebereichen,
- Beschreibung der *intrarelationalen Abhängigkeiten* (Festlegen der Schlüsselattribute), gegebenenfalls Bestimmung von *Functional Dependencies* (FDs) und *Multivalued Dependencies* (MVD),
- Beschreibung der *interrelationalen Abhängigkeiten* (*Inclusion Dependencies* (IDs)),
- Beschreibung weiterer Integritätsbedingungen.

Eine FD $X \rightarrow Y$ (lies: X bestimmt Y funktional; X, Y sind Mengen von Attributen) liegt vor, wenn für alle Relationen R' eines Relationenschemas R gilt: zwei Tupel, deren Komponenten in X übereinstimmen, stimmen auch in Y überein [RA97, S. 8-3]. D.h., zu jedem X-Wert oder X-Wertetupel in R' gibt es höchstens einen Y-Wert oder Y-Wertetupel [BA96, S. 711]. Werden nun FDs festgelegt, können damit die möglichen Relationen auf bestimmte gültige Relationen eingeschränkt werden. Fälle, in denen ein X-Wert oder ein X-Wertetupel mehrere Y-Werte oder Y-Wertetupel funktional bestimmt, wobei alle diese Kombinationen voneinander unabhängig sind, werden MVDs genannt.

Unter IDs wird in [BA96, S.712] die Formulierung der Fremdschlüsselbedingung verstanden: Zu jedem Wert (wenn es nicht der Null-Wert ist) eines Fremdschlüsselattributs einer Relation R_2 muß ein gleicher Wert des Primärschlüssels in irgendeinem Tupel von Relation R_1 vorhanden sein [RA97, S. 4-6].

Führt man die o.g. Teilaufgaben unter Beachtung der Regeln für die Transformation eines ER-Modells in ein logisches Schema [BA96, S.716 ff.] für den Entity-Typ *Auftrag* aus, so erhält man folgendes (noch unvollständiges) logisches Schema, wobei die für die Formulierung von interrelationalen Abhängigkeiten notwendigen Relationsschemata namentlich aufgeführt sind:

LOGIC SCHEMA

Name:
ASV-System

Relationsschemata:
Auftrag, Anzeige, Benutzer, Rubrik, Unterrubrik, ...

Integrity Constraints[7]*:*
Auftrag[Kundennummer] $\subseteq$ Benutzer[Zugangsnummer]
Auftrag[Anzeigennummer] $\subseteq$Anzeige[Anzeigennummer]

[7] Inclusion Dependencies (IDs) werden als spezielle Integrity Constraints aufgefaßt.

Auftrag[Rubriknummer] ⊆ Rubrik[Rubriknummer]
Auftrag[Unterrubriknummer] ⊆ Unterrubrik[Unterrubriknummer]
Auftrag[Verwalternummer] ⊆ Benutzer[Zugangsnummer]
...
weitere:
Für eine Anzeige (Anzeigennummer) dürfen höchstens ein Veränderungsauftrag und höchstens ein Verlängerungsauftrag existieren;
...

Nachfolgend das konkrete Relationsschema für den Entity-Typ *Auftrag*:

RELATIONSSCHEMA

Name:	Auftrag		
Attribute:	Auftragsnummer:	integer	not null
	Neuauftrag:	char(1)	
	Verlaengerungsauftrag:	char(1)	
	Veraenderungsauftrag:	char(1)	
	Kundennummer:	integer	not null
	Anzeigennummer:	integer	
	abgeschlossen:	char(1)	
	Auftragsdatum:	date	not null
	StartHTMLSeite:	charakter(30)	
	Ueberschrift:	charakter(80)	
	Schlagwoerter:	charakter(100)	
	Kurzbeschreibung:	charakter(255)	
	KontaktEmail:	charakter(80)	
	Rubriknummer:	integer	
	Unterrubriknummer:	integer	
	Schaltbeginn:	date	
	Schaltende:	date	
	Verlaengerungsende:	date	
	Verwalternummer:	integer	
Key:	Auftragsnummer		
FD:	[Unterrubriknummer] -> [Rubriknummer]		
Notes:	Ein Auftrag kann nur entweder ein Neuauftrag oder ein Veränderungsauftrag oder ein Verlängerungsauftrag sein. Zulässige Einträge in den Attributen *Neuauftrag*, *Veraenderungsauftrag*, *Verlaengerungsauftrag* und *abgeschlossen* sind ‚0' und ‚1'. ...		

Das vollständige logische Schema mit allen Relationsschemata befindet sich in Anhang D.

3.1.2 Normalisierung

Intrarelationale Abhängigkeiten können zu Redundanzen und bei Änderungsoperationen zu Integritätsverletzungen führen. Mittels der Normalisierung kann diese Redundanz vermieden werden, indem jedes Relationenschema gegebenenfalls in neue Schemata, die *Basisrelationen-Schemata* genannt werden, zerlegt wird. Dabei werden nichtredundante Funktionalabhängigkeiten eines Relationenschemas erhalten [RA97, S.8-5]. Die Zerlegung muß dabei in jedem Fall den verlustfreien Verbund garantieren. Durch den Normalisierungsprozeß werden aus nicht normalisierten Relationenschemata neue Relationenschemata gewonnen, die einer bestimmten *Normalform* genügen. Man unterscheidet dabei folgende Normalformen: die *Non-first Normal-Form* (NF^2), die *1NF, 2NF, 3NF, 4NF, 5NF* sowie die *Boyce/Codd-Normalform* (BCNF).

Laut [BA96, S. 727] gelten als Kriterien für einen guten relationalen Datenbankentwurf:

1. erreichte Normalform
2. verlustloser Verbund
3. Gewährleistung der Abhängigkeiten

Das 2. und 3. Kriterium sind gewährleistet, wenn die Relationenschemata in 3NF vorliegen. Bei höheren Normalformen ist dies nicht immer der Fall. Daher wurde die Normalisierung für die Relationenschemata des ASV-Systems bis zur 3NF durchgeführt. Die theoretischen Grundlagen wurden dabei [RA97, S. 8-6ff.] entnommen:

Ein Relationenschema ist in NF^2, falls es „Attribute“ enthält, die wiederum Relationen sind. Dies ist für kein Relationenschema des ASV-Systems (vgl. Anhang D) der Fall, d.h., sie befinden sich bereits mindestens in 1NF.

Ein Relationenschema R ist in 2NF, wenn es in 1NF ist und jedes *Nicht-Primärattribut* von R voll funktional von jedem Schlüsselkandidaten in R abhängt. Dabei ist ein Primärattribut ein Attribut, das zu mindestens einem Schlüsselkandidaten des Schemas gehört. Wie man überprüfen kann, ist dies für alle Relationenschemata des ASV-Systems der Fall.
Aber auch bei Relationenschemata in 2NF kann es aufgrund von transitiven Abhängigkeiten noch zu Änderungsanomalien kommen. Eine Attributmenge Z des Relationenschemas R ist transitiv abhängig von einer Attributmenge X in R, wenn gilt:

FD $X \rightarrow Y$, FD $Y \rightarrow Z$, nicht FD $Y \rightarrow X$, und Z ist keine Teilmenge von Y.

Damit kann man nun die 3NF definieren: Ein Relationenschema R befindet sich in 3NF, wenn es sich in 2NF befindet und jedes Nicht-Primärattribut von R von keinem Schlüsselkandidaten von R transitiv abhängig ist.

Bei der Erarbeitung des logischen Schemas wurde bereits festgestellt, daß in den Relationenschemata *Auftrag* und *Anzeige* die FD `Unterrubriknummer` $\rightarrow$ `Rubriknummer` gilt. Es würde daher gelten

- FD `Auftragsnummer` (bzw. `Anzeigennummer`) → `Unterrubriknummer` und
- FD `Unterrubriknummer` → `Rubriknummer`.

Somit wäre `Rubriknummer` transitiv abhängig von `Auftragsnummer` (bzw. `Anzeigennummer`). Das bedeutet, daß sich die Relationenschemata *Auftrag* und *Anzeige* nicht in 3NF befinden. Da bereits das Relationenschema *Unterrubrik* die Zuordnung einer Rubrik zu einer Unterrubrik abbildet, wäre eine Lösung die Streichung des Attributes *Rubrik* aus *Auftrag* und *Anzeige*. Damit wäre es aber nicht möglich, eine Anzeige nur für eine Rubrik und nicht gleichzeitig für eine Unterrubrik dieser Rubrik zu schalten. Dies kann durchaus wünschenswert sein, wenn die Unterteilung in Unterrubriken erst mit wachsender Anzeigenanzahl schrittweise eingeführt werden soll. Zum anderen führt der Zugriff auf die Rubrik einer Anzeige, wenn einmal viele Tupel gespeichert sind, zu längeren Antwortzeiten, da für den Zugriff die Tabellen *Anzeige, Unterrubrik* und *Rubrik* miteinander verbunden werden müßten.

Daher fiel die Entscheidung, die beiden Relationenschemata *Auftrag* und *Anzeige* nicht in die 3NF zu überführen.

3.1.3 Views und Zugriffsrechte

Nach abgeschlossener Normalisierung besteht die Möglichkeit, mittels *Views* (Sichten) einen wertabhängigen Zugriffsschutz und mittels *Zugriffsrechten* die Autorisierung einzelner Benutzer für den Zugriff auf diese Sichten und Relationen einzurichten.

Da beim ASV-System nur ein einzelner Nutzer, nämlich das ASV-Programm, neben dem Datenbankadministrator (DBA) zugelassen werden soll und die Benutzer (Kunden, Verwalter, Nutzer) des ASV-Systems keinen direkten Zugriff auf die Datenbank erhalten sollen, wurde keine Definition von Sichten durchgeführt. Das ASV-Programm erhält alle Rechte, die zum Betrieb des ASV-Systems notwendig sind. Dies ist unproblematisch, solange man gewährleistet, daß keine direkte und ungewollte Manipulation der Datenbank durch Benutzer stattfinden kann. Insbesondere benötigt die Anwendung das Schreibrecht auf die Tabellen *Auftrag*, *Anzeige*, *geschaltet_Anzeige*, *geschaltet_Auftrag* und *Sitzung*. Es mag am Anfang sinnvoll erscheinen, für die Tabellen *Rubrik*, *Unterrubrik*, *Stadt*, *Benutzer* nur ein Leserecht zu vergeben. Für zukünftige Erweiterungen des ASV-Systems, die zum Beispiel das Einrichten neuer Kunden oder das Einrichten von neuen Rubriken und Unterrubriken vom Verwaltungsteilsystem aus gestatten sollen, müßte aber auch hier ein Schreibrecht vergeben werden.

Damit ist der Datenbankentwurf abgeschlossen. Das in Anhang D beschriebene logische Schema des ASV-Systems ist zugleich das in der Implementierungsphase zu verwendende Datenbankschema.

3.2 Sicherheitsaspekte

Da der WWW-Server des ASV-Systems an das Internet angeschlossen ist, muß mit Angriffen gerechnet werden. Die erste Angriffsmöglichkeit ergibt sich dabei aus der Tatsache, daß der Datenverkehr zwischen einem WWW-Server und einem Browser standardmäßig unverschlüsselt erfolgt. Angreifer, die an geeigneter Stelle im Netzwerk sitzen, können somit gesendete Daten lesen, was insbesondere bei übergebenen Paßwörtern problematisch ist, oder die Daten sogar abfangen und manipulieren, ohne daß die Empfänger, WWW-Server oder WWW-Browser, etwas davon bemerken.

Weitere Angriffsmöglichkeiten ergeben sich aufgrund der gewählten Realisation als formularbasierte Anwendung durch die Möglichkeit, daß Kunden und Verwalter versuchen könnten, direkt in den Steuerungsfluß der Anwendung einzugreifen oder bereits entgegengenommene und geprüfte Daten zu verändern.

Ebenfalls Beachtung müssen Problemstellungen finden, die die korrekte Konfiguration der verwendeten Software betreffen. Das ASV-System ist eine WWW-Anwendung, die laut Pflichtenheft auch auf virtuellen WWW-Servern von WWW-Providern zum Einsatz kommen soll. Das bedeutet, daß ein unbekannter Nutzerkreis über das WWW auf das ASV-System zugreifen wird, welches auf einem Server installiert ist, zu dem möglicherweise eine große Anzahl von Personen Kommandozeilenzugriff haben. In der Regel wird der Betreiber des ASV-Systems auf die Konfiguration des Betriebssystems und des WWW-Servers keinen Einfluß haben. Diese Aufgabe fällt dem Administrator des WWW-Servers zu, der dabei auf umfangreiche Literatur wie [CB96] und [EI98] zurückgreifen kann. Der Betreiber des Anzeigendienstes wird dagegen auf die korrekte Konfiguration von PHP und des DBMS achten müssen.

3.2.1 Observation und Manipulation des Datenverkehrs

Vor Beginn einer Sitzung müssen Kunden und Verwalter ihr Paßwort in ein Formular eingeben und an das ASV-System senden. Werden keine geeigneten Maßnahmen getroffen, erfolgt die Übermittlung der Paßwörter unverschlüsselt. Ein Angreifer könnte nun versuchen, diese Paßwörter an geeigneter Stelle im Klartext herauszufiltern und sich so unberechtigterweise Zugang zum ASV-System zu verschaffen.

Ein weiteres Problem ist, daß ein Browser standardmäßig nicht feststellen kann, ob eine abgerufene Seite wirklich von dem WWW-Server stammt, von dem sie abgefordert wurde, oder ob ein feindlicher Server diese Seite nicht etwa emuliert, um so ebenfalls an die Paßwörter zu gelangen.

Weiterhin sollte auch eine Manipulation der übermittelten Inhalte vermieden oder zumindest erkannt werden können. Insbesondere bei kommerziell genutzten Realisierungen des ASV-Systems könnten manipulierte Produktdaten oder Preise großen Schaden für die Kunden verursachen.

Eine Lösung für diese Problemstellungen bietet das von Netscape entwickelte *Secure Socket Layer Protocol* (SSL-Protokoll)[8].

Im Gegensatz zu anderen Ansätzen wie dem *S-HTTP*-Protokoll von *EIT* ist SSL nicht allein auf das WWW beschränkt, sondern kann auch für andere Internet-Dienste wie FTP und TELNET genutzt werden. Ein großer Vorteil von SSL ist, daß es für die WWW-Anwendung selbst transparent ist, d.h., man kann eine sichere Verbindung nutzen, ohne Veränderungen an der eigenen Applikation vornehmen zu müssen. Nach [OH99] und [EI98, S. 385] hat sich das SSL-Protokoll bisher am weitesten durchgesetzt und wird von den Standardbrowsern (Netscape Navigator und Microsoft Explorer) unterstützt.

Das SSL-Protokoll weist nach [EI98, S.386] folgende Eigenschaften auf:

- Eine Verbindung ist privat, wobei nach einem initialen Verbindungs-Handshake ein Sitzungsschlüssel vereinbart wird, der für ein symmetrisches Verschlüsselungsverfahren nach *RC4* oder *DES* verwendet wird, um die übertragenen Daten zu verschlüsseln.
- Eine Authentifikation von Server und Client wird durch ein *Public-Key*-Verfahren ermöglicht.
- Eine Verbindung ist zuverlässig, da Veränderungen an den übertragenen Daten durch einen *Message Authentication Code* (MAC) aufgedeckt werden können. Für die Erzeugung des MAC kommt der *MD5*-Algorithmus zum Einsatz.

Die Funktionsweise des SSL-Protokolls sei im folgenden kurz beschrieben[9]:

Netscape nutzt *RSA Public-Key-Kryptographie*, die von der *RSA Data Security, Inc.* lizensiert wurde und die für Authentifikation und Verschlüsselung genutzt wird. Public-Key-Kryptographie ist dabei eine Technik, die ein Paar *asymmetrischer Schlüssel* für Verschlüsselung und Entschlüsselung benutzt. Jedes Paar von Schlüsseln besteht dabei aus einem *Public-Key* und einem *Private-Key*. Der Public-Key wird dabei allen Interessenten bekannt gemacht, während der Private-Key streng geheim gehalten wird. Nachrichten, die mit dem Private-Key verschlüsselt wurden, können nur mit dem Public-Key entschlüsselt werden. Umgekehrt können Daten, die mit dem Public-Key verschlüsselt wurden, nur mit dem Private-Key wieder entschlüsselt werden.

Ein Problem bei der Kommunikation über Datennetze besteht darin, die Authentizität des Kommunikationspartners zu verifizieren. Der dabei stattfindende Prozeß (Authentifikation) überprüft die Identität, so daß ein Teilnehmer sicher sein kann, daß sein Gegenüber derjenige ist, für den er sich ausgibt. Die dabei auftretenden Problemstellungen werden im folgenden anhand eines Beispieles schrittweise dargestellt.

Die Notation `{Nachricht}Key` bedeutet, daß die `Nachricht` mit dem `Key` verschlüsselt oder entschlüsselt wird.

---

[8] Freier, A. O. u.a.: *The SSL Protocol Version 3.0,* http://home.netscape.com/eng/ssl3/draft302.txt, 1996

[9] Netscape*: How SSL works*, http://developer.netscape.com/tech/security/ssl/howitworks.html

Angenommen, A möchte B authentifizieren. B überläßt dafür A seinen Public-Key. A sendet B nun eine Nachricht `{Nachricht}`, die B mit seinem Private-Key verschlüsselt und wieder an A zurücksendet:

`{Nachricht}Bs_Private_Key`.

A kann nun mit Bs Public-Key diese Nachricht entschlüsseln und mit der ursprünglich an B gesendeten vergleichen. Stimmen beide überein, weiß A, daß er mit B spricht.

Für B hat dieses Verfahren allerdings den Nachteil, daß er eine Nachricht verschlüsselt, über deren Bedeutung er sich vielleicht nicht ganz im klaren ist. Nach der Verschlüsselung und Übersendung an A kann er jedoch nicht mehr abstreiten, der Absender zu sein, da ja nur er in Besitz seines Private-Keys ist.

Ein erster Schritt, um dieses Problem zu lösen, ist die Einführung eines *Message-Digests* und die Verschlüsselung dieses Message-Digests anstelle der Nachricht. Dieser Vorgang wird *Signierung* genannt. Ein Message-Digest verfügt über folgende Eigenschaften: Der Originalinhalt ist nur sehr schwer aus einem Digest herleitbar und es ist kaum möglich, eine andere Nachricht zu generieren, die den gleichen Digest erzeugt.

B könnte nun aus der von A erhaltenen Nachricht einen Digest erzeugen und diesen verschlüsseln:

`{Digest[Nachricht]}Bs_Private_Key`.

A kann nun diese Daten entschlüsseln, seinerseits einen Digest der Originalnachricht erstellen und die beiden Digests vergleichen. B sollte aber, um sich zu schützen, die signierten Daten selbst erzeugen. Der Kommunikationsablauf könnte nun folgendermaßen aussehen:

A → B	Bist Du B?
B → A	Ja, ich bin B; `{Digest[Ja, ich bin B]}Bs_Private_Key.`

Dabei sendet B auf As Anfrage eine selbstgenerierte Nachricht im Klartext und danach den verschlüsselten Digest. So kann A feststellen, daß B in der Tat B ist und B muß keine Nachricht signieren, über deren Bedeutung er sich möglicherweise nicht im klaren ist.

Noch gibt es aber folgendes Problem: Wie kann B seinen Public-Key an A übermitteln, ohne daß folgendes passiert:

A -> C	Hallo
C -> A	Hallo A, ich bin B und hier ist mein Cs_Public_Key
A -> C	Beweise es!
C -> A	Ja, ich bin B; `{Digest[Ja, ich bin B]}Cs_Private_Key.`

Alles, was C braucht, um vorzuspielen, er sei B, ist ein beliebiges Paar asymmetrischer Schlüssel.

Um dieses Problem zu lösen wurden *Zertifikate* eingeführt. Ein Zertifikat hat folgenden Inhalt: Den Namen des Ausstellers; das Subjekt (Person, Unternehmen), für welches das Zertifikat ausgestellt wurde; den Public-Key des Subjekts; verschiedene Zeitstempel. Das Zertifikat wird vom Aussteller mit seinem Private-Key signiert, wobei der Public-Key des Ausstellers allgemein bekannt ist. Bei den Ausstellern der Zertifikate wiederum handelt es sich in der Regel um staatlich anerkannte Behörden oder Unternehmen.

Es ergibt sich folgender neuer Kommunikationsablauf:

A -> B	Hallo
B -> A	Hallo A, ich bin B und hier ist mein `Bs_Zertifikat`
A -> B	Beweise es!
B -> A	Ja, ich bin B; `{Digest[Ja, ich bin B]}Bs-Private-Key.`

A kann nun Bs Zertifikat auf Echtheit überprüfen, indem er die Signatur mit dem Public-Key der Zertifizierungsstelle entschlüsselt, den Digest erstellt und mit dem entschlüsselten Digest vergleicht. Ist das Zertifikat echt, kann A den angegebenen Subjektnamen lesen und den im Zertifikat enthaltenen Public-Key von B nutzen, um den mit Bs Private-Key verschlüsselten Digest zu entschlüsseln. Nun braucht A nur noch den Digest selbst zu erstellen und mit dem entschlüsselten zu vergleichen. Stimmen beide überein, ist B wirklich B.

Nachdem B von A authentifiziert werden konnte, kann A eine Geheimnachricht schicken, indem A die Geheimnachricht mit Bs Public-Key verschlüsselt. So kann nur B diese Nachricht entschlüsseln, indem er seinen Private-Key benutzt. Diese Geheimnachricht kann nun zum Beispiel der Schlüssel für einen symmetrischen Kryptographiealgorithmus sein wie *DES* oder *RC4*. A kennt den Schlüssel, da A ihn erzeugt hat, B kennt ihn, da er ihn mit seinem Private-Key entschlüsseln konnte. Nun können A und B verschlüsselt miteinander kommunizieren.

Es gibt jedoch ein weiteres Problem: Ein Angreifer C kann sich zwischen die Kommunikation von A und B einklinken und die Nachrichten verfälschen, nachdem die Geheimnachricht ausgetauscht wurde. Dabei kann C zwar nicht den Inhalt der Nachrichten verstehen, die im folgenden mittels symmetrischer Verschlüsselung ausgetauscht werden, aber er kann durch seine Verfälschung, falls er damit eine valide Nachricht erzeugt, A oder B täuschen, da sich A und B darauf verlassen, daß die Nachricht gesichert ist und von seinem authentifizierten Gegenüber stammt, und danach handeln.

Die Lösung für dieses Problem ist die Verwendung eines MAC. Der MAC ist ein Digest, der aus der zu übermittelnden Nachricht und der Geheimnachricht besteht.

```
MAC := Digest[Nachricht, Geheimnachricht]
```

Da der Angreifer C die `Geheimnachricht` nicht kennt, ist die Wahrscheinlichkeit gering, daß er den korrekten MAC errät. Bei einer 128-bit MAC-Länge,

generiert mit dem *MD5*-Verfahren (ebenfalls eine Entwicklung von RSA), besteht laut Netscape eine Wahrscheinlichkeit von 1 zu 18.446.744.073.709.551.616, daß der MAC erraten werden kann.

Mit Hilfe des MAC ergibt sich folgender Kommunikationsablauf:

A -> B	Hallo
B -> A	Hallo A, ich bin B und hier ist mein `Bs-Zertifikat`
A -> B	Beweise es!
B -> A	Ja, ich bin B, `{Digest[Ja, ich bin B]}Bs_Private_Key`
A -> B	O.k., hier ist der Geheimschlüssel: `{Geheimnachricht}Bs_Public_Key`
B -> A	`{Nachricht, MAC}Geheimnachricht`

Sollte C die Nachricht nun verfälschen, würde das bemerkt werden und A und B könnten die Kommunikation abbrechen.

Für den Apache Web-Server ist eine Implementierung des SSL-Protokolls als SSL-Patch verfügbar. Mittels der frei verfügbaren Bibliothek *ssleay*[10] können Schlüssel und sogar eigene Zertifikate erzeugt werden. Die Installation und Konfiguration wird ausführlich in [EI98, S. 387-405] dargestellt. Bei der Verwendung dieser selbsterstellten Zertifikate muß der Nutzer jedoch vor Aufnahme der Verbindung dieses Zertifikat ausdrücklich akzeptieren, was durch einen Browser-Dialog geschieht. Für den kommerziellen Einsatz empfiehlt sich daher der Kauf eines Zertifikates, das von einer anerkannten Zertifizierungsstelle ausgestellt wird. Eine Übersicht über anerkannte Zertifizierungsstellen ist in [EI98, S. 396] enthalten.

Bei der Nutzung von SSL wird der Betreiber des Anzeigendienstes auf den WWW-Provider angewiesen sein, da dieser den Betrieb des Apache-SSL-Servers gestatten muß.

3.2.2 Problemstellungen aufgrund der Realisierung als formularbasierte Anwendung

1. Problemstellung: Eingriff in den Steuerungsfluß

Das bereits mehrfach erwähnte Problem von formularbasierten Anwendungen, die mittels CGI realisiert werden, ist, daß eine CGI-Anwendung nach der Erzeugung eines HTML-Formulars und seiner Übertragung an den Browser beendet wird und damit der aktuelle Zustand verloren geht. Um diesen Zustand für die nachfolgende Bearbeitung zu speichern, muß er im zurückgegebenen HTML-Formularen abgelegt werden[11]. Wird dieses HTML-Formular nun wiederum an den WWW-Server zurückgeschickt, liest die CGI-Anwendung die Zustandsvariablen aus und

[10] http://www.ssleay.org

[11] Die im folgenden aufgezeigten Problemstellungen treten analog auch bei der Verwendung alternativer Methoden zur Zustandsspeicherung wie *Cookies* oder *URL-Kodierung* [LO98] auf.

kann daraus die nächste auszuführende Operation ableiten. Da der Quellcode eines HTML-Formulars im Klartext lesbar ist, könnte ein Angreifer diese Zustandsinformationen studieren und schließlich benutzen, um eine veränderte Abfolge der HTML-Formulare zu bewirken oder um HTML-Formulare zu überspringen. So könnte er von der Upload-Seite aus direkt die Speicherung des Auftrages in der Datenbank auslösen, ohne die HTML-Formulare für die Anzeigen- und Schaltdaten passiert zu haben.

Problemlösung

Die mutwillige Veränderung der Steuerflußdaten an sich ist nicht zu verhindern. Allerdings könnte das ASV-System so gestaltet werden, daß es nur bestimmte Steuerflußdaten akzeptiert, in Abhängigkeit vom zuletzt eingenommenen Zustand z_n. Dazu müßte die Information über z_n nicht nur in der zurückgegebenen HTML-Seite, sondern auch in einer Datei oder in der Datenbank, die beide nicht vom Benutzer manipulierbar sind, abgelegt werden. Zum Beispiel als zusätzliches Attribut in der Sitzungstabelle. Weiterhin müßte eine Systemsteuertabelle angelegt werden, in die die zulässigen Paare $[z_n, z_{n+1}]$ eingetragen werden. Bei jedem Aufruf könnte jetzt anhand des gespeicherten Zustandes z_n und der Systemsteuertabelle überprüft werden, ob die Einnahme des neuen geforderten Zustandes zulässig ist.

In der Beispielrealisation des ASV-Systems kann auf eine Implementierung dieser Steuerflußkontrolle verzichtet werden, da der Schaden, den ein Kunde oder Verwalter durch diese Art von Manipulation anrichten kann, begrenzt ist:
Er könnte im schlimmsten Fall versuchen, die Annahme eines unvollständigen Auftrages zu erwirken, was aber vor der Abspeicherung der Auftragsdaten erkannt werden kann.

2. Problemstellung: Nachträgliches Ändern von Daten

Auch diese Problemstellung erwächst aus der Möglichkeit, von außen in den Zustand des ASV-System einzugreifen. Überprüft das System so zum Beispiel den Inhalt eines bestimmten Eingabefeldes und speichert es danach diese Variable im zurückgegebenen HTML-Formular, um sie für die endgültige Abspeicherung vorzumerken, so kann ein Angreifer dieses HTML-Formular verändern und so zum Beispiel den Inhalt der Variable löschen, was das System, falls es keine nochmalige Überprüfung vornimmt, nicht bemerken würde.

Problemlösung

Auch hier könnte man die Übergabe von Variablen von Formular zu Formular durch eine Zwischenspeicherung in der Datenbank vermeiden. Dies würde jedoch bedeuten, daß man die Antwortzeit der Anwendung durch die zusätzlichen Datenbankzugriffe erhöht. Ein wiederholter Test vor dem endgültigen Abspeichern ist dagegen eine weitaus einfacher zu realisierende Lösung, die auch die Antwortzeit kaum beeinträchtigt. Daher kann auf die Zwischenspeicherung der Daten verzichtet werden.

3. Problemstellung: Versuch, fremde Sitzungen zu übernehmen

Die Verwaltung von Sitzungen wurde im Abschnitt 2.2.1 eingeführt, um feststellen zu können, ob die Bearbeitung eines noch nicht abgeschlossenen Auftrages abgebrochen wurde oder aber zu einem Kunden oder Verwalter gehört, der aktuell eingeloggt ist und den Auftrag damit nur noch nicht ordnungsgemäß beendet hat. Die Einführung von Sitzungen erhöht aber auch die Sicherheit des ASV-Systems. Zum einen wird so bei unverschlüsselten Verbindungen die Gefahr des Abhörens des Passwortes *vermindert*, indem das Passwort nur einmal zu Beginn der Sitzung unverschlüsselt übertragen und nach erfolgter Überprüfung des Passwortes durch das ASV-System nur die Sitzungsnummer von Formular zu Formular weitergereicht wird. Die Sitzungsnummer wird aber nach Beendigung der Sitzung ungültig. Ein Angreifer, der die Datenübertragung nach der Übermittlung des Passwortes abfängt, kann also nur für die Zeitdauer diese Sitzung Schaden anrichten, was natürlich trotzdem sehr unangenehm sein kann.

Indem nicht das Passwort an sich von Formular zu Formular übergeben wird, kann zum anderen vermieden werden, daß ein Angreifer den Festplatten-Cache seines Vorbenutzers untersucht und dort das gespeicherte HTML-Formular mit dem gespeicherten Passwort findet.

Während bei einer unverschlüsselten Verbindung das Abfangen des Passwortes oder der Sitzungsnummer nicht vermieden werden kann, sollte bei einer verschlüsselten Verbindung die Übernahme einer laufenden Sitzung durch einen Angreifer verhindert oder aber sehr erschwert werden. Die Übernahme einer fremden Sitzung könnte erfolgen, wenn die Sitzungsnummer für den Angreifer erratbar ist.

Problemlösung

Die erfolgreiche Übernahme einer Sitzung setzt voraus, daß sowohl die Sitzungsnummer des Opfers bekannt ist als auch seine Benutzernummer. Zudem muß der Angriff erfolgen, wenn das Opfer gerade eingeloggt ist. Dies alles macht einen Angriff unwahrscheinlich. Zusätzlich sollte man die Sitzungsnummer möglichst schwer erratbar gestalten. Dies kann zum Beispiel durch die Erzeugung einer großen Zufallszahl geschehen. PHP bietet durch die Funktion *UniqId()* eine andere Möglichkeit, indem basierend auf der aktuellen Zeit in Mikrosekunden eine bis zu 114 Zeichen lange Sitzungsnummer generiert werden kann, die einmalig ist, auch wenn auf einem anderen System die gleiche Funktion zur gleichen Zeit in Mikrosekunden aufgerufen werden würde.

All dies wird jedoch hinfällig, falls der Angreifer den Datenverkehr unverschlüsselt abhören kann.

4. Problemstellung: Löschen fremder Dateien

Das ASV-System soll bei Anzeigenannahme und Veränderungsaufträgen das Löschen einmal hochgeladener Dateien des Kunden ermöglichen. Ein Angreifer könnte nun versuchen, die Löschung von fremden Dateien zu bewirken, indem er ein Formular emuliert und etwa folgenden Dateinamen übergibt: „../*“. Dieses

Argument an einen (Unix)-Löschbefehl übergeben, würde die Löschung aller Dateien im übergeordneten Verzeichnis bedeuten.

Problemlösung

Argumente, die Dateinamen enthalten, müssen immer auf die Zeichen „..“ und „/“ untersucht werden. Sollten diese Zeichen in Dateinamen vorkommen, so muß von einem Angriff ausgegangen und die Bearbeitung abgebrochen werden. Eine Nachricht an den Betreiber des Anzeigendienstes sollte über diesen Angriff informieren.

5. Problemstellung: Versteckte Unix-Kommandos

Das ASV-System wird das File-System von Unix nutzen, um die Anzeigendateien zu speichern und gegebenenfalls wieder zu löschen. Das bedeutet, daß sichergestellt werden muß, daß in den übergebenen Argumenten wie Dateinamen keine versteckten Unix-Shell-Kommandos einfügt sind. Ein Angreifer könnte sonst ein Formular emulieren und als zu löschenden Dateinamen in der Anzeigenannahme zum Beipiel „harmlos.gif; rm –rf ~/*“ angeben, womit er alle Dateien des Anzeigendienstbetreibers löschen könnte, falls dieses Argument an ein Unix-Kommando übergeben wird.

Problemlösung

PHP bietet die Funktion *EscapeShellCmd($unsicheres_Argument)*, die alle Zeichen entwertet, die genutzt werden könnten, um versteckte Kommandos auszuführen. Alternativ dazu kann man aber auch spezielle PHP-Funktionen für Dateioperationen nutzen wie zum Beispiel *Unlink()* zum Löschen von Dateien, die versteckte Unix-Kommandos nicht ausführen.

3.2.3 Problemstellungen hinsichtlich der Konfiguration des DBMS und PHP

Das ASV-Gesamt-System ist einer Reihe von Sicherheitsrisiken ausgesetzt, die folgende Bereiche umfassen:

- Angriffe von der Kommandozeile des WWW-Servers aus
- Angriffe über das Internet

Falls für das ASV-System ein eigener Server im Unternehmen oder der Institution des Betreibers des Anzeigendienstes zur Verfügung steht, auf dem nur das ASV-System installiert ist und auf den nur der Betreiber des Anzeigen-Online-Dienstes bzw. bestimmte autorisierte Mitarbeiter Zugriff haben, kann von Angriffen über die Kommandozeile praktisch keine Gefahr für das ASV-System und die in ihm gespeicherten Daten ausgehen. In der Regel werden jedoch auf den WWW-Server des ASV-Systems eine Vielzahl von Personen Zugriff haben, die auch zu einem anderen Unternehmen oder einer anderen Institution gehören können. Dies genau ist nämlich bei der Anmietung von Speicherplatz auf einem virtuellen WWW-Server eines WWW-Providers der Fall. Im folgenden werden die einzelnen möglichen Angriffe analysiert und Lösungsmöglichkeiten dargestellt.

Dabei wird vorausgesetzt, daß das ASV-System auf einem Unix-Betriebssystem installiert wird und die im Pflichtenheft genannten Software-Produkte zum Einsatz kommen. Auch wenn es nicht ausgeschlossen ist, daß die verwendeten Software-Produkte (Betriebssystem, DBMS, Programmiersprache) Programmfehler aufweisen, so sollte doch deren sichere und korrekte Funktionsweise vorausgesetzt werden.

1. Problemstellung: Auslesen und Manipulation der Datenbank

Das DBMS des ASV-Systems wird als eigenständige Anwendung auf dem Server installiert. Es verfügt über ein Kommandozeilen-Interface, mit dem auf den Inhalt der ASV-Datenbank zugegriffen werden kann, sowohl mit DDL- als auch mit DML-Befehlen. Wenn ein Angreifer Zugang zur Datenbank erhält, kann er demnach sowohl die Struktur als auch den Inhalt der Datenbank nach Belieben ändern. Da DBMS über bestimmte Ports (an denen der Datenbank-Dämon lauscht) oder Treiber (zum Beispiel ODBC) einen Zugriff auf die Datenbank auch von außen ermöglichen, besteht ebenso die Möglichkeit, daß ein Angreifer versuchen könnte, sich von einem anderen Computer aus Zugang zu den gespeicherten Daten zu verschaffen.

Problemlösung

Das DBMS muß so konfiguriert werden, daß nur der Betreiber des Anzeigendienstes bzw. sein DBA und das ASV-System an sich Zugang erhalten können. Um die Möglichkeit eines Angriffs auf die Datenbank von einem anderen Computer aus einzuschränken, sollten die *Hosts*, von denen auf das DBMS zugriffen werden darf, genau spezifiziert werden. Die konkrete Konfiguration des DBMS für das ASV-System wird in Abschnitt 4.1.1 beschrieben.

2. Problemstellung: Ausführung von fremdem PHP-Code

Wie das DBMS wird auch PHP als eigenständige Anwendung installiert, sofern es nicht als Modul in den WWW-Server hineinkompiliert wird. PHP muß also auch *für alle* zur Ausführung freigegeben werden, da WWW-Server in der Regel als User *nobody*, *noone* etc. laufen und damit keinen Zugriff auf private Dateien eines User haben. Da PHP als Parameter übergebene Dateien nach PHP-Befehlen durchsucht und diese dann ausführt, könnte ein Angreifer versuchen, PHP mit selbst geschriebenen PHP-Dateien aufzurufen. Dies an sich würde nur die Mitnutzung des installierten PHP bedeuten und wäre nicht gefährlich, solange PHP als User *nobody* ausgeführt wird.

Bereits jetzt steht aber fest, daß PHP mit der *UserID* des ASV-Systems ausgeführt werden muß, um die vorgesehenen Dateioperationen (Speicherung und Löschung von Anzeigendateien auf dem Server) ausführen zu können.
Wenn ein Angreifer nun PHP mit eigenem Code aufrufen kann, würde er durch die umfassenden Betriebssystemroutinen von PHP einen vollständigen Zugang zu allen Dateien des ASV-Systems sowohl lesend als auch schreibend erhalten.

Problemlösung

Es kann nicht verhindert werden, daß Angreifer PHP im CGI-Verzeichnis des Servers ausführen können, da die Ausführung wie oben beschrieben *für alle* möglich sein muß. PHP muß demnach so konfiguriert werden, daß nur bestimmte PHP-Dateien ausgeführt werden können. Zum Beispiel nur PHP-Dateien, die sich in einem bestimmten Verzeichnis des ASV-System-Accounts befinden. Dies ist möglich und wird in Abschnitt 4.1.2 beschrieben.

3. Problemstellung: Ausspionieren des PHP-Quellcodes

PHP-Dateien enthalten PHP-Code in Klarform, der bei der Ausführung von PHP interpretiert wird. Damit müssen PHP-Dateien vor dem Lesezugriff durch Angreifer geschützt werden.

Problemlösung:

Die Ausführung von PHP mit der UserID des ASV-Systems ermöglicht es, die PHP-Quellcodedateien nur für das ASV-System lesbar zu machen. Somit wird anderen Usern kein Zugriff ermöglicht (siehe auch Abschnitt 4.1.2).

3.2.4 Abschließende Bemerkungen

Wie gezeigt wurde, kann ein Großteil potentieller Sicherheitsrisiken durch eine sorgfältige Konfiguration der benutzten Software-Systeme wie PHP und MySQL ausgeschlossen werden.

Weiterhin kann durch die Einführung einer Sitzungsverwaltung die Gefahr *vermindert* werden, daß bei einer unverschlüsselten Verbindung das Passwort abgehört wird. Auszuschließen ist dies bei einer unverschlüsselten Verbindung jedoch nicht. Gegen das Abhören der übertragenen Daten (z.B. des Passwortes) kann man sich bei einer formularbasierten Lösung durch die Verwendung des SSL-Protokolls schützen. Die Verwendung von SSL setzt jedoch (zumindest bei der Verwendung des Apache-Webservers) die Unterstützung des Administrators des WWW-Servers voraus.

Ein grundsätzliches Mißtrauen allen Benutzereingaben gegenüber schränkt die Möglichkeiten für einen Angreifer, Schaden anzurichten, zusätzlich ein. Die Möglichkeit, in den Daten- und Kontrollfluß der CGI-Anwendung einzugreifen, ist und bleibt jedoch ein Sicherheitsrisiko, dem je nach Anforderung durch eine mehrfache Überprüfung der entgegengenommenen Daten (bei der Eingabe, vor der Speicherung) oder durch eine persistente Zustandsspeicherung auf Serverseite (Sitzungszustand, Eingabedaten) begegnet werden kann.

4. Implementierung

Bevor mit der Implementierung des ASV-Systems begonnen werden kann, muß die Entwicklungsumgebung, bestehend aus DBMS und Programmiersprache, installiert und konfiguriert werden. Ebenfalls vor der konkreten Umsetzung steht die Ausarbeitung eines Implementierungsfahrplans.

4.1 Konfiguration von MySQL und PHP

Im Pflichtenheft des ASV-Systems wurde vereinbart, daß der Betreiber des ASV-Systems seine Software auch auf dem Server eines Providers installieren können muß. Für einen privaten Betreiber des ASV-Systems wird dies die Regel sein. Da aber auch bei der Installation in Institutionen der Betreiber meist keinen Einfluß auf die Installation und Konfiguration des WWW-Servers hat und dabei auf den Systemverwalter angewiesen ist, wird im folgenden nur die Konfiguration der Datenbank MySQL und der Scriptsprache PHP beschrieben werden. Wie im Abschnitt 3.2 ausgeführt wurde, muß die Konfiguration dieser Software-Pakete sorgfältig vorgenommen werden, um Sicherheitslücken zu vermeiden. Die *Installation* beider Produkte ist ausführlich in den Quellcode-Distributionen[12] beschrieben.

4.1.1 Konfiguration von MySQL

Die Datenbank MySQL muß vor PHP installiert werden, damit die MySQL-Bibliothek in PHP eingebunden werden kann. Nach erfolgreicher Installation sind nun folgende Schritte notwendig, um MySQL für das ASV-System zu konfigurieren:

- Schließen der Sicherheitslöcher,
- Anlegen der ASV-Datenbank,
- Vergabe von Nutzerrechten für die Datenbank des ASV-Systems.

4.1.1.1 Schließen der Sicherheitslöcher

Bei der Installation wird mittels des Scripts `mysql_install_db` ein Nutzer *root* für das DBMS eingerichtet, der über alle Rechte verfügt, für den aber kein Passwort eingetragen wurde. Folge: Jeder, der auf dem Server ein Login hat, könnte sich unbeschränkten Zugriff auf jede vorhandene Datenbank verschaffen. Daher sollte als erstes das *root*-Passwort eingerichtet werden. Dazu wechselt man in das Installationsverzeichnis von MySQL und ruft im Unterverzeichnis `/bin` den MySQL-Client `mysql` mit den Parametern `-u root mysql` auf. Damit erhält man als Nutzer *root* Zugriff auf die Datenbank *mysql*, in der alle Nutzer und deren Rechte verwaltet werden. Danach verfährt man folgendermaßen:

```
mysql >  SET PASSWORD FOR root=PASSWORD('neues_passwort');
```

[12] Verfügbar unter *http://www.php.net* bzw. *http://www.mysql.net*

Bei Arbeiten mit der Kommandozeilenversion `mysql` von MySQL als Nutzer *root* sollte man nun immer folgendermaßen vorgehen:

```
shell > mysql –u root –p gewuenschte_Datenbank
```

MySQL fragt daraufhin das Passwort interaktiv ab. Niemals jedoch sollte man das Passwort schon auf der Kommandozeile eingeben, da mittels des Unix-Befehls *ps* die Kommandozeilenargumente angezeigt werden. Also nicht:

```
shell > mysql –u root –p root_passwort gewuenschte_Datenbank
```

Standardmäßig ist für den Nutzer *root* als Host, von dem aus eine Verbindung zur Datenbank aufgebaut werden kann, *localhost* angegeben. Dies ist ausreichend, da der Betreiber des ASV-Systems über ein Login auf dem Server verfügt. Diese Einstellung sollte nicht geändert werden, um Versuche, das Passwort von anderen Rechnern aus zu erraten, von vornherein auszuschließen.

Weiterhin sollte man den MySQL-Dämon `mysqld` bei Verwendung des bereitgestellten Scripts `mysql.server` zum Starten und Stoppen des MySQL-Dämons nicht als Unix-Nutzer *root* laufen lassen[13]. Dies geschieht durch Editieren des Scriptes `mysql.server`. Hier kann man als neuen Unix-Nutzernamen das Login des ASV-Systembetreibers angeben.

Zuletzt sollte man die Unix-Zugriffsrechte aller MySQL-Verzeichnisse auf *700* setzen, damit nur der Betreiber des ASV-Systems und das unter seinem Login laufende MySQL Zugriffe auf die Datenbankverzeichnisse haben.

4.1.1.2 Anlegen der Datenbank

Bevor die DDL-Statements für die Erzeugung des Datenbankschemas eingelesen werden können, muß die Datenbank für das ASV-System angelegt werden. Dies geschieht mittels des Programms `mysqladmin`:

```
shell > mysqladmin –u root –p create asv
```

Dabei steht `asv` für den Namen der Datenbank des ASV-Systems.

4.1.1.3 Vergabe von Nutzerrechten für die Datenbank des ASV-Systems

Als letzter Schritt bei der Konfiguration des DBMS muß nun noch ein Nutzer eingerichtet werden, der Zugriff auf die Datenbank `asv` hat. Dabei sollten diesem Nutzer aber nur die Rechte gegeben werden, die für den Betrieb des ASV-Systems benötigt werden. Zum Beispiel sollte der Nutzer nicht das Recht erhalten, Datenbanken zu „droppen“. Bei der Einrichtung des Nutzers kann ebenfalls bestimmt werden, von welchen Hosts aus die Verbindung aufgenommen werden kann. Aus Sicherheitsgründen sollte dies nur vom Host *localhost* aus möglich sein. Dies ist ausreichend, da die ASV-Anwendung auf dem gleichen Server installiert wird wie das DBMS.

---

[13] Die Unix-Nutzerbezeichnung *root* steht nicht in Beziehung mit dem Nutzer *root* der Datenbank.

Die konkrete Anweisung, mit der auch gleich der Datenbank-Nutzer `asv` angelegt wird, lautet:

```
mysql > GRANT SELECT,INSERT,UPDATE,DELETE
      > ON asv.* TO asv@'localhost'
      > IDENTIFIED BY 'asv-passwort';
```

4.1.2 Konfiguration von PHP

Die Konfiguration von PHP beginnt bereits vor der Installation mit Aufruf des Scripts `setup`. Hier ist neben der Vereinbarung, daß PHP mit MySQL-Support kompiliert werden soll, das Setzen des *safe mode* notwendig, um später ein *Document-Root*-Verzeichnis vereinbaren zu können, außerhalb dessen keine PHP-Dateien gelesen und interpretiert werden. Die Vereinbarung *force cgi-redirection* ist für das ASV-System nicht zu setzen, da keine WWW-Server-Zugriffkontrollen verwendet werden sollen.

Nach erfolgreicher Installation müssen nun die in Abschnitt 3.2 besprochenen Sicherheitslücken geschlossen werden:

- Verhinderung der Ausführung von fremdem PHP-Code
- Verhinderung der Einsichtnahme in den PHP-Code des ASV-Systems

Die Ausführung von fremdem PHP-Code kann man durch die Vereinbarung des Parameters *doc_root* in der Datei `php.ini` verhindern. Setzt man beispielsweise *doc_root= /home/asv-betreiber/asv-system/*, so werden PHP-Dateien nur gelesen und ausgeführt, die im Verzeichnis *asv-system/* oder dessen Unterverzeichnissen liegen.

Um die Einsichtnahme in den PHP-Code zu verhindern, könnte man nun ein Verzeichnis als doc_root vereinbaren, das außerhalb des WWW-Verzeichnisses des WWW-Servers liegt. Damit wäre die Möglichkeit ausgeschlossen, die PHP-Datei über den Browser direkt als HTML-Datei zu laden, d.h., daß folgende Anfrage zu einer *Document not found*-Meldung des WWW-Servers führen würde:

http://www.asv-system.de/php_datei.php3 anstelle von
*http://www.asv-system.de/*cgi-bin/php*/php_datei.php3*.

Leider wird dadurch jedoch noch nicht ausgeschlossen, daß ein Angreifer, der ein Login auf dem Rechner besitzt, die PHP-Dateien ausliest: Bisher müssen sie nämlich für alle lesbar sein, da PHP, wenn es durch den WWW-Server aufgerufen wird, mit dessen Unix-UserID läuft, was in der Regel ein Nutzer ohne spezielle Leserechte ist. Dieses Problem kann man jedoch umgehen, indem man PHP mit den Unix-Nutzerrechten des Betreibers des ASV-System ausführt. Dies kann man mit der Unix-Anweisung

```
shell> chmod u+=rwxs
```

erreichen. Nun ist es möglich, das Verzeichnis mit den PHP-Dateien nur für den Betreiber des ASV-Systems lesbar zu machen, zum Beispiel mittels der Anweisung

```
shell> chmod 700 /home/asv-betreiber/asv-system
```

Weder über das WWW noch über ein Login auf dem Server ist es nun möglich, den Inhalt der PHP-Dateien im Klartext zu lesen.

4.2 Implementierungsfahrplan

Um bei der Implementierung effizient vorgehen zu können, muß auch die Reihenfolge der Teilschritte festgelegt werden, die nacheinander realisiert und nach ihrer Erst-Implementierung ausführlich getestet werden, bevor man den nächsten Schritt im Implementierungsfahrplan umsetzt.

Im folgenden soll ein Ablaufplan für eine effiziente Implementierung entworfen werden. Kritisch könnte angemerkt werden, daß vorgesehen ist, bestimmte Funktionen der Auftragsannahme (Pseudocodefunktionen 2.2 bis 2.12) und Auftragsverwaltung (Pseudocodefunktionen 3.5 bis 3.7) jeweils schrittweise während des Implementierungsprozesses zu ergänzen und nicht auf einmal zu implementieren. Dies geschieht jedoch aus der Überzeugung heraus, daß es für die Stabilität und Korrektheit der Anwendung wichtig ist, daß der Programmierer in einer Etappe immer einen ganzen Prozeß (z. B. die Annahme von Neuaufträgen) umsetzt und testet. So müssen Tests nicht separiert für eine Prozeßkomponente (z. B. das Anlegen eines Neu-, Veränderungs- oder Verlängerungsauftrages in der Datenbank oder das Löschen eines abgebrochenen Neu-, Veränderungs- oder Verlängerungsauftrages) durchgeführt werden, sondern können über den ganzen Prozeß hinweg geführt werden.

1. Implementierungsschritt : Einrichten der Datenbank
Pflichtenheftfunktionen : -
Pseudocodefunktionen : -

Das im Anhang D angegebene Datenbankschema ist hierbei in die MySQL-spezifische DDL zu überführen. Ergebnis sollte eine `load`-Datei sein, die alle DDL-Vereinbarungen enthält, um die Datenbank für das ASV-System einzurichten. Hierbei könnte es auch notwendig sein, bestimmte Tabellen schon mit Einträgen zu versehen. Zum Beispiel könnten in der Tabelle *Benutzer* bereits ein Beispielkunde und ein Verwalter eingetragen werden. Desweiteren muß eine `dump`-Datei geschrieben werden, die den Datenbankinhalt für ein Backup sichert und schließlich eine `drop`-Datei, die es ermöglicht, die Datenbank vor einer erneuten Anwendung der `load`-Datei in ihren Ausgangszustand zu versetzen.

2. Implementierungsschritt : Hilfsfunktionen
Pflichtenheftfunktionen : -
Pseudocodefunktionen : -

Die Implementierung der Hilfsfunktionen umfaßt die Kapselung der Funktionen für den Datenbankzugriff sowie die Realisierung von Funktionen, die implizit in Pseudocodefunktionen enthalten sind, zum Beispiel: Erzeugung einer Sitzungsnummer.

3. Implementierungsschritt : Sitzungsverwaltung
Pflichtenheftfunktionen : /F10/, /F120/, /F210/
Pseudocodefunktionen : 1.1 bis 1.6
Die Umsetzung der Sitzungsverwaltung umfaßt alle Funktionen, die im Pseudocode unter *Verwalte Sitzung* ausgeführt sind, d.h. insbesondere die Funktionen zur Erzeugung einer neuen Sitzung, zur Überprüfung von übergebenen Sitzungsdaten und zum Löschen eines Sitzungseintrages. Nach diesem Implementierungsschritt sollte das Ein- und Ausloggen über das WWW bereits simulierbar sein.

4. Implementierungsschritt : Servicearea
Pflichtenheftfunktion : /F130/
Pseudocodefunktion : 2.1
Die Servicearea ist der Bereich, in den der Kunde nach dem Einloggen gelangt. Wichtige Funktionen sind hierbei: Die Darstellung von bereits geschalteten Anzeigen, die erst einmal per Hand in die unter 1. eingerichtete Datenbank eingetragen werden können, mit möglicher Funktionsauswahl (Ändern, Verlängern, Löschen) und Funktionen zum Erkennen von bestehenden Aufträgen, die die mögliche Funktionsauswahl einschränken.

5. Implementierungsschritt : Schalten von Neuaufträgen
Pflichtenheftfunktionen : /F20/ bis /F110/
Pseudocodefunktionen : 2.2 bis 2.12
Nachdem die Datenbank eingerichtet, Sitzungen ermöglicht und die Servicearea gestaltet worden ist, kann mit der Programmierung der Anzeigenannahme begonnen werden. Sie umfaßt folgende Funktionsgruppen: Anlage eines neuen Auftrages, File-Upload, Löschung hochgeladener Dateien, Voransicht, Annahme der Anzeigendaten, Annahme der Schaltdaten, Ausgabe der Übersichtsseite, Auftragsabschluß.

6. Implementierungsschritt : Verwaltungsarea
Pflichtenheftfunktionen : /F220/ bis /F230/
Pseudocodefunktionen : 3.1 bis 3.4
Analog zur Servicearea ist die Verwaltungsarea der Bereich, in dem der Verwalter Statistiken und den Überblick über eingegangene Aufträge erhält. Zu implementieren sind folglich Funktionen zur Erzeugung der Statistiken, zur Erzeugung der Übersichten von Neuaufträgen, Veränderungs- und Verlängerungsaufträgen und zum Erkennen von Aufträgen, die bereits von anderen Verwaltern bearbeitet werden. Ebenfalls kann hier bereits das Sperren von Aufträgen, sobald ein Verwalter die Bearbeitung übernimmt, realisiert werden.

7. Implementierungsschritt : Freischalten von Neuaufträgen
Pflichtenheftfunktionen : /F240/ bis /F320/
Pseudocodefunktionen : 3.5 bis 3.7
Das Freischalten von Neuaufträgen kann auf die im 5. Implementierungsschritt bereits realisierten Pseudocodefunktionen 2.3 bis 2.10 aufbauen. Es ist jedoch notwendig, die Funktionalität so zu erweitern, daß bereits gespeicherte Werte ange-

zeigt werden können. Neu muß die Funktion 3.7 implementiert werden, die einen Auftrag aus der Auftragstabelle in die Anzeigentabelle der Datenbank transferiert, wenn der Verwalter entscheidet, daß dieser Auftrag freigeschaltet werden soll.

8. Implementierungsschritt : Ein- und Aussetzen von Anzeigen
Pflichtenheftfunktion : /F205/
Pseudocodefunktion : 2.13
Das Ein- und Aussetzen einer Anzeige ist eine mit kleinem Aufwand zu implementierende Funktion, die aber für einen sinnvollen Betrieb eines Anzeigendienstes notwendig ist und deshalb gleich nach der Annahme und Bearbeitung von Neuaufträgen realisiert werden sollte.

9. Implementierungsschritt : Schalten von Veränderungsaufträgen
Pflichtenheftfunktionen : /F140/ bis /F200/
Pseudocodefunktionen : 2.2 bis 2.12
Analog zum 5. Implementierungsschritt unter Nutzung der im 7. Implementierungsschritt gemachten Veränderungen der Pseudocodefunktionen 2.3 bis 2.10. Allerdings müssen bestimmte Beschränkungen hinsichtlich der zur Veränderung angebotenen Werte implementiert werden. So soll es nicht möglich sein, Datumsangaben zu ändern.

10. Implementierungsschritt : Freischalten von Veränderungsaufträgen
Pflichtenheftfunktionen : /F330/ bis /F410/
Pseudocodefunktionen : 3.5 bis 3.7
Veränderungsaufträge unterscheiden sich in der Anzeigenverwaltung von Neuaufträgen nur in dem Punkt, daß in der Anzeigentabelle noch ein Eintrag mit der unveränderten Anzeige vorhanden ist und die Pseudocodefunktion 3.7 so ergänzt werden muß, daß sie dies berücksichtigt.

11. Implementierungsschritt : Schalten von Verlängerungsaufträgen
Pflichtenheftfunktion : /F203W/
Pseudocodefunktionen : 2.2, 2.9 bis 2.12
Keine Anmerkungen.

12. Implementierungsschritt : Freischalten von Verlängerungsaufträgen
Pflichtenheftfunktionen : /F420W/ bis /F500W/
Pseudocodefunktionen : 3.5 bis 3.7
Keine Anmerkungen.

13. Implementierungsschritt : Anzeigendarstellung
Pflichtenheftfunktionen : /F510/ bis /F540/
Pseudocodefunktionen : 4.1 bis 4.3
Die Anzeigendarstellung wird als letzter Implementierungsschritt umgesetzt. Hierbei müssen keine Änderungsoperationen, sondern nur noch Leseoperationen realisiert werden.

5. Ergebnisdokumentation Beispielrealisation

Anhand von Screenshots soll im folgenden eine Beispielrealisation des ASV-Systems vorgestellt werden. Dabei wird der gesamte Anzeigenannahme- und Verwaltungsprozeß Schritt für Schritt dargestellt, um einen Eindruck von der Arbeitsweise der Applikation zu vermitteln. Bei der Beispielrealisation handelt es sich um einen fiktiven Anzeigen-Online-Dienst, der es Kunden ermöglicht, Produktpräsentationen in den Rubriken Hard- und Software, Finanzdienstleistungen, Immobilien etc. zu veröffentlichen.

Die Screenshots befinden sich am Ende des vorliegenden Abschnittes.

5.1 Anzeigenannahme

Die Annahme einer Anzeige erfolgt in folgenden Schritten:
1. Einloggen in das ASV-System mit Kundennummer und Passwort
2. Upload der Anzeigendateien mit Voransicht
3. Eingabe der Anzeigen- und Schaltdaten
4. Bestätigung aller Daten

5.1.1 Einloggen in das ASV-System

Klickt man auf der Startseite (siehe Abb. 10) auf den Link *Service*, gelangt man zur Login-Seite für die Kunden (siehe Abb. 11). Für die Demonstration wurde ein Kundenaccount eingerichtet. Kundennummer und Kundenpaßwort sind in die entsprechenden Felder einzugeben und danach der *Weiter*-Button anzuklicken.

Wurden Kundennummer und Passwort korrekt eingegeben, so gelangt man zur Servicearea (siehe Abb. 12). Ist dem Kunden ein Fehler unterlaufen, so erhält er eine Fehlermeldung mit der Aufforderung, zur Login-Seite zurückzukehren und den Anmeldeprozeß zu wiederholen. Da der Beispielkunde noch keine Anzeige aufgegeben hat, ist die Anzeigenübersicht leer.

Am Anfang und Ende jeder Serviceseite befindet sich ein *Exit*-Link, der ein ordnungsgemäßes Ausloggen ermöglicht. Ordnungsgemäß bedeutet dabei, daß die aktuelle Sitzung für diesen Kunden durch Löschen des Sitzungseintrages aus der Sitzungstabelle beendet wird. Bricht der Kunde dagegen die Sitzung nicht ordnungsgemäß ab, indem er etwa einen anderen Bookmark anwählt, so wird seine Sitzung automatisch nach 30 Minuten Inaktivität gelöscht. Dies ist ein Kompromiß zwischen Sicherheit und Komfort für den Kunden, der zum Beispiel bei der Voransicht seiner Anzeige einen Rechtschreibfehler bemerkt hat, die Korrektur an der HTML-Seite vornimmt und sich danach nicht gleich einem Auto-Logout ausgesetzt sehen will.

Um die erste Anzeige aufzugeben, klickt man auf den Link *Anzeigenannahme* und gelangt so zur Upload-Seite.

5.1.2 Die Anzeigendateien uploaden

Die Uploadseite bietet die Möglichkeit, alle zu einer Anzeige gehörenden HTML-Dateien, GIF-, JPEG-Grafiken, JAVA-Applets, Sound- und Videodateien in das ASV-System nacheinander hochzuladen. Dabei werden alle Dateien in ein für diese Anzeige reserviertes Verzeichnis geschrieben, so daß Links relativ auf das gleiche Verzeichnis zu setzen sind.
Zum Beispiel: *<a href="logo.gif">* anstatt von *<a href="../pfad/logo.gif">*. Darüber wird der Kunde im Infotext der Uploadseite informiert (siehe Abb. 13).

Für die Demonstration wurde eine Anzeige gewählt, die aus HTML-Dateien, GIF-Grafiken und einem JPEG-Bild besteht.

Die Auswahl der Datei, die hochgeladen werden soll, erfolgt durch Betätigen des *Durchsuchen*-Buttons oder durch direkte Eingabe des Dateinamens in das Eingabefeld (siehe Abb. 13). Bei Betätigung des *Durchsuchen*-Buttons erscheint ein Auswahlfenster mit dem Inhalt der Festplatte. Man kann nun die Datei, die hochgeladen werden soll, durch einen Mausklick ausgewählen und durch einen weiteren Klick auf *OK* bestätigen. Pro Upload-Durchgang kann dabei jeweils nur eine Datei ausgewählt werden (siehe auch Abschnitt 1.2 und 2.3.2 in diesem Kapitel). Im Beispiel soll die Datei *topselframe.htm* als erste Datei hochgeladen werden. Dies geschieht durch die Betätigung des *Upload ausführen*-Buttons.
Zur Bestätigung des erfolgreichen Uploads erscheint auf einer neuen Formularseite die Ausschrift *Die Datei ‚topselframe.htm' wurde hochgeladen*. Die Datei erscheint ebenfalls im Verzeichnis aller bisher hochgeladenen Dateien (siehe Abb. 14).
Auf die gleiche Weise kann man nun nacheinander auch die anderen zur Anzeige gehörenden Dateien in das ASV-System hochladen.

Sobald die erste Datei hochgeladen ist, erweitert sich die Upload-Seite um die Möglichkeit, eine bereits hochgeladene Datei wieder zu löschen (siehe Abb. 15). Diese Möglichkeit erlaubt es dem Kunden bei einem Neuauftrag, versehentlich hochgeladene Anzeigen wieder zu löschen. Bei der Auftragsänderung dagegen kann der Kunde dann nicht mehr benötigte Anzeigen löschen.
Sobald man eine HTML-Seite hochgeladen hat, kann man die Voransichts-funktion nutzen. Dazu wählt man diese HTML-Seite aus und kann dann durch Wahl der Voransichtsfunktion mittels Radio-Button (siehe Abb. 16) und Betätigung des *Weiter*-Buttons die HTML-Seite betrachten. Mit der Voransicht ist es möglich, die Funktionalität aller Links und das korrekte Laden aller Grafiken, Applets etc. zu überprüfen. Wurden noch nicht alle zur Anzeige gehörenden Dateien hochgeladen, so erscheinen die entsprechenden Grafiken und Bilder natürlich als *Broken Links* bzw. die Servermeldung *Document not found*. In diesem Fall kann man zur Uploadseite zurückkehren und den Upload fortsetzen. Muß eine HTML-Seite korrigiert werden, kann man die – zuvor in einem externen Editor – korrigierte Version ein weiteres Mal hochladen. Die bereits bestehende Datei wird dabei automatisch überschrieben.

Sind alle zur Anzeige gehörenden Dateien hochgeladen und hat man die Anzeige per Voransicht überprüft, kann man durch Auswahl der Option *Weiter zum Auftragsformular* per Radio-Button und dem Betätigen des *Weiter*-Buttons (siehe Abb. 16) zur Eingabe der Anzeigen- und Schaltdaten gelangen.

5.1.3 Eingabe der Anzeigen- und Schaltdaten

Das Auftragsformular dient zur Eingabe der Anzeigen- und Schaltdaten *Überschrift (Anzeigeneintrag), Rubrik, Bundesland, Schaltbeginn, Schaltdauer*. Damit wurde nur ein Teil der in der Konzeption vorgesehenen Anzeigendaten realisiert, der aber ausreicht, um die Funktionsweise zu demonstrieren. Daher wurden auch die Formularseiten für die Anzeigen- und Schaltdaten zu einer Formularseite zusammengefaßt. Anstelle der Möglichkeit, Städte als Strukturierungskriterium anzugeben, wurden in der Beispielrealisation Bundesländer gewählt, was aber kein prinzipieller Unterschied ist. Für die Dokumentation wurde für die Anzeige die Überschrift *Eine Testanzeige*, die Rubrik *Hard- und Software*, das Bundesland *Sachsen*, als Schaltbeginn der *15.06.1999* und die Schaltdauer von *einer Woche* gewählt (siehe Abb. 17 und Abb. 18).

Mit diesen Eingaben ist der Annahmeprozeß fast beendet. Durch Auswahl der Option *Weiter zur Auftragserstellung* per Radio-Button und dem Betätigen des *Weiter*-Buttons gelangt der Kunde zur nächsten Formularseite, die noch einmal die zur Anzeige gehörenden Dateien und Daten in einer Übersicht darstellt.

5.1.4 Bestätigung der Daten

Auf der Bestätigungsseite (siehe Abb. 19) sind noch einmal die hochgeladenen Dateien sowie Anzeigen- und Schaltdaten aufgelistet. Stimmen alle Angaben, so kann der Kunde durch Auswahl der Option *Auftrag bestätigen* und dem Betätigen des *Weiter*-Buttons die Anzeigenannahme abschließen (die Optionswahl ist auf der Abbildung nicht zu sehen). Das ASV-System bestätigt daraufhin die erfolgreiche Anzeigenannahme. Dabei wird dem Kunden auch die für diese Anzeige vergebene Anzeigennummer mitgeteilt (siehe Abb. 20).

Durch einen Klick auf den *Zur Anzeigenübersicht*-Button gelangt der Kunde nun wieder zur Startseite der Servicearea. Dort sieht er nun seinen Auftrag verzeichnet. Da die Freischaltung noch nicht erfolgt ist, wird dem Kunden mitgeteilt, daß die Anzeige in Bearbeitung ist (siehe Abb. 21).

Die Anzeigenannahme kann nun ordnungsgemäß durch das Ausloggen mittels *Exit*-Link beendet werden.

5.2 Anzeigenverwaltung

Durch einen Klick auf den Link *Verwaltung* (zum Beispiel auf der Startseite der Beispielrealisation) gelangt der Verwalter zur Login-Seite der Anzeigenverwaltung (siehe Abb. 22).
Durch Eingabe seiner Verwalternummer und seines Verwalterpasswortes erhält der Verwalter Zutritt zur Verwaltungsarea (siehe Abb. 23). Die Verwaltungsarea bietet sowohl einen Überblick über die Anzahl der eingegangenen Aufträge (die eben geschaltete Anzeige ist bereits als *Neuaufträge: 1* verzeichnet) als auch die Möglichkeit, direkt zur Bearbeitung der jeweiligen Auftragsart überzugehen.

Auch in der Servicearea ermöglichen es die *Exit*-Links, sich ordnungsgemäß auszuloggen.

Durch Auswahl von *Neuaufträge anzeigen* gelangt man zur Übersicht aller neu eingegangenen Aufträge, die (wie in der Konzeption der Benutzungsoberfläche gefordert) tabellarisch dargestellt werden (siehe Abb. 24). Die in der Konzeption der Benutzungsoberfläche besprochene Möglichkeit, Aufträge direkt freizuschalten bzw. zu verwerfen, wurde in der Beispielrealisation nicht umgesetzt, da sie sich von der *Freischalt-* bzw. *Verwerf*funktion in der Detailansicht (siehe dort) nicht unterscheiden.
Statt dessen ermöglicht es ein *Bearbeiten*-Button, zur Detailansicht des Auftrages zu wechseln (siehe Abb. 25). Hier erhält der Verwalter wie gefordert einen Gesamtüberblick über alle Kunden-, Anzeigen- und Schaltdaten und kann nach einer Voransicht entscheiden, ob er den Auftrag freischaltet, verwirft (löscht) oder zur späteren Bearbeitung zurückstellt (siehe Abb. 26).
Durch die Auswahl der Option *Freischalten* per Radio-Button und Betätigen des *Weiter*-Buttons wird der Auftrag freigeschaltet und das ASV-System übernimmt automatisch die Anzeige zum gewünschten Schaltbeginn für die gewünschte Schaltdauer.

5.3 Anzeigenansicht

Sobald der Schaltbeginn der freigeschalteten Anzeige erreicht wird, ist sie für den Nutzer sichtbar. Über die Startseite gelangt der Nutzer mittels Links zu der gewünschten Rubrik (im Beispiel die Rubrik *Hard- und Software*, für die eine Anzeige geschaltet wurde). Gemäß der Festlegungen im Abschnitt 2.3.2 werden erst einmal nur die Anzeigen aufgelistet, die nicht für ein spezielles Bundesland, also *bundesweit*, geschaltet wurden. (Da hier aber keine weiteren Anzeigen geschaltet wurden, ist in Abb. 27 auch keine *bundesweit* geschaltete Anzeige sichtbar.) Bei Auswahl des gewünschten Bundeslandes (im Beispiel *Sachsen*) mittels der rechten Auswahlbox wird dann aber die geschaltete Beispielanzeige sichtbar (siehe Abb. 28). Durch einen Klick auf den Anzeigeneintrag kann die Anzeige nun betrachtet werden.
Da in der Beispielrealisation die in der Konzeption vorgesehene Eingabe einer Kurzbeschreibung und die Eingabe von Schlagwörtern bei der Anzeigenannahme nicht implementiert wurden, kann die Suche nach Schlagwörtern und die Anzeige der Kurzbeschreibung in der Demonstration auch nicht gezeigt werden.

5.4 Anzeigenservice

Loggt sich der Kunde nach dem Freischalten seines Auftrages in die Servicearea ein, sieht er seinen Auftrag nun wie in Abb. 29 dargestellt. Er kann nun die Funktionen *verändern*, *verlängern* und *aussetzen* nutzen. Die Funktion *aussetzen* hat dabei den Effekt, daß die Anzeige, wie gefordert, für den Nutzer sofort (d.h. ohne zusätzliches Freischalten durch den Verwalter) nicht mehr sichtbar ist. Die Funktion *verändern* führt den Kunden durch die gleichen Formulare wie bei der Annahme des Neuauftrages. Wie gefordert, werden dabei die Anzeigen- und Schaltdaten des Auftrages in den jeweiligen Eingabefeldern angezeigt und können editiert werden. Bei einem Verlängerungsauftrag muß der Kunde dagegen nur das in Abb. 30 dargestellte Formular ausfüllen.

Das Freischalten von Veränderungs- und Verlängerungsaufträgen erfolgt analog zum Freischalten von Neuaufträgen.

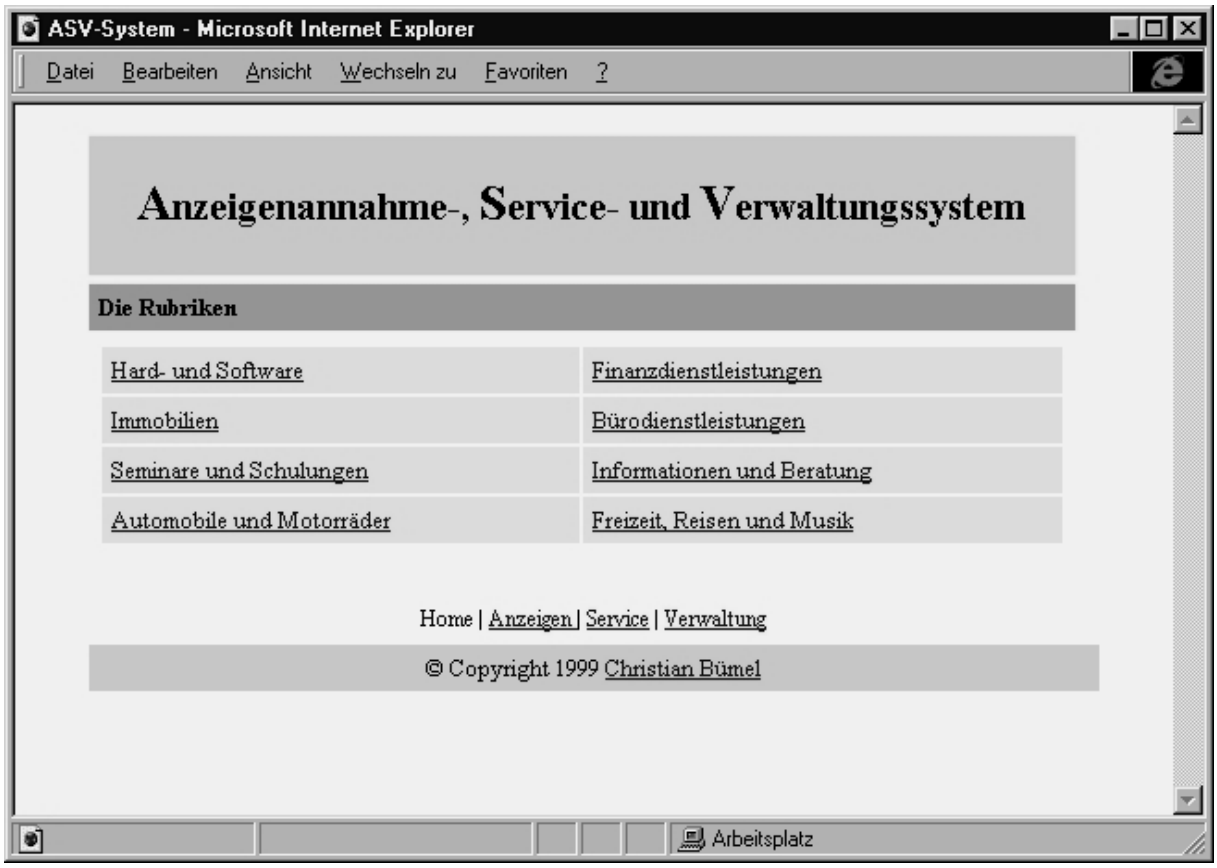

Abb. 10 Startseite der Beispielrealisation des ASV-Systems

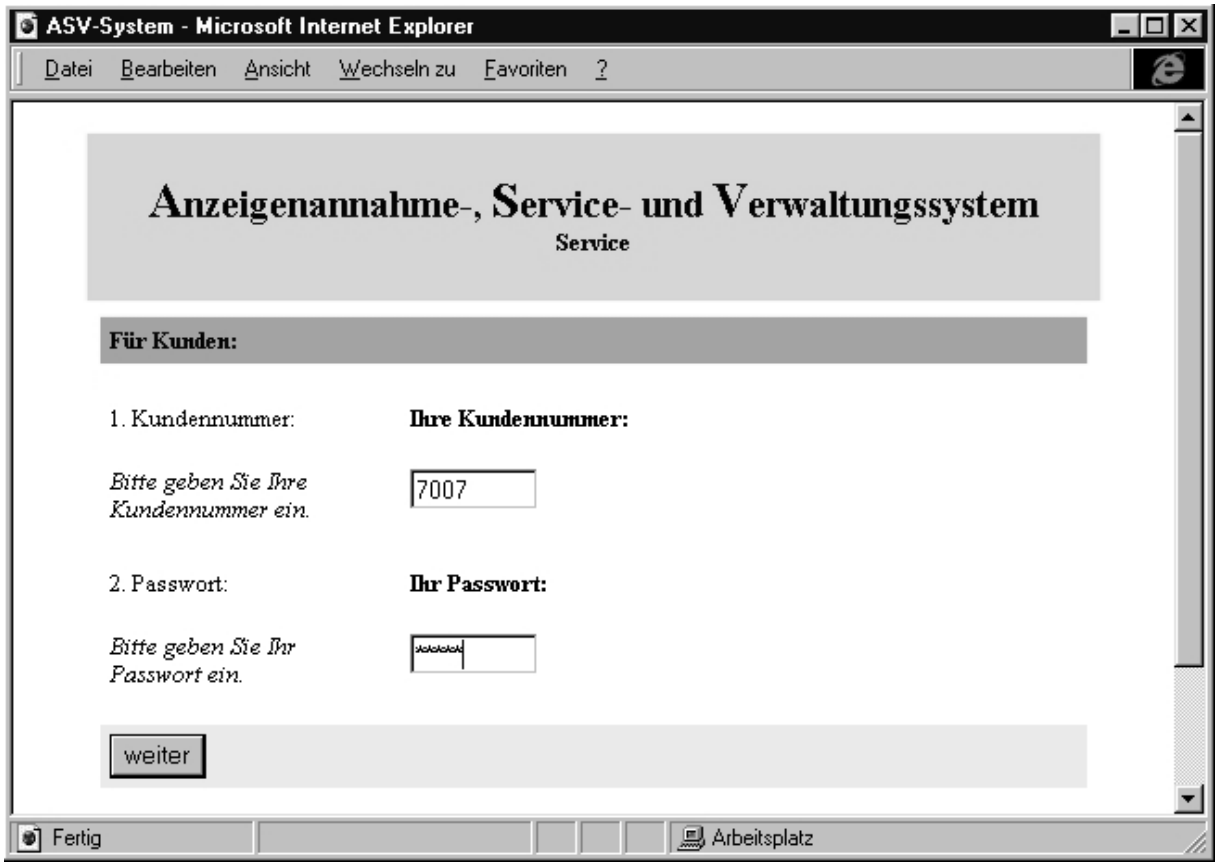

Abb. 11 Die Login-Seite für Kunden

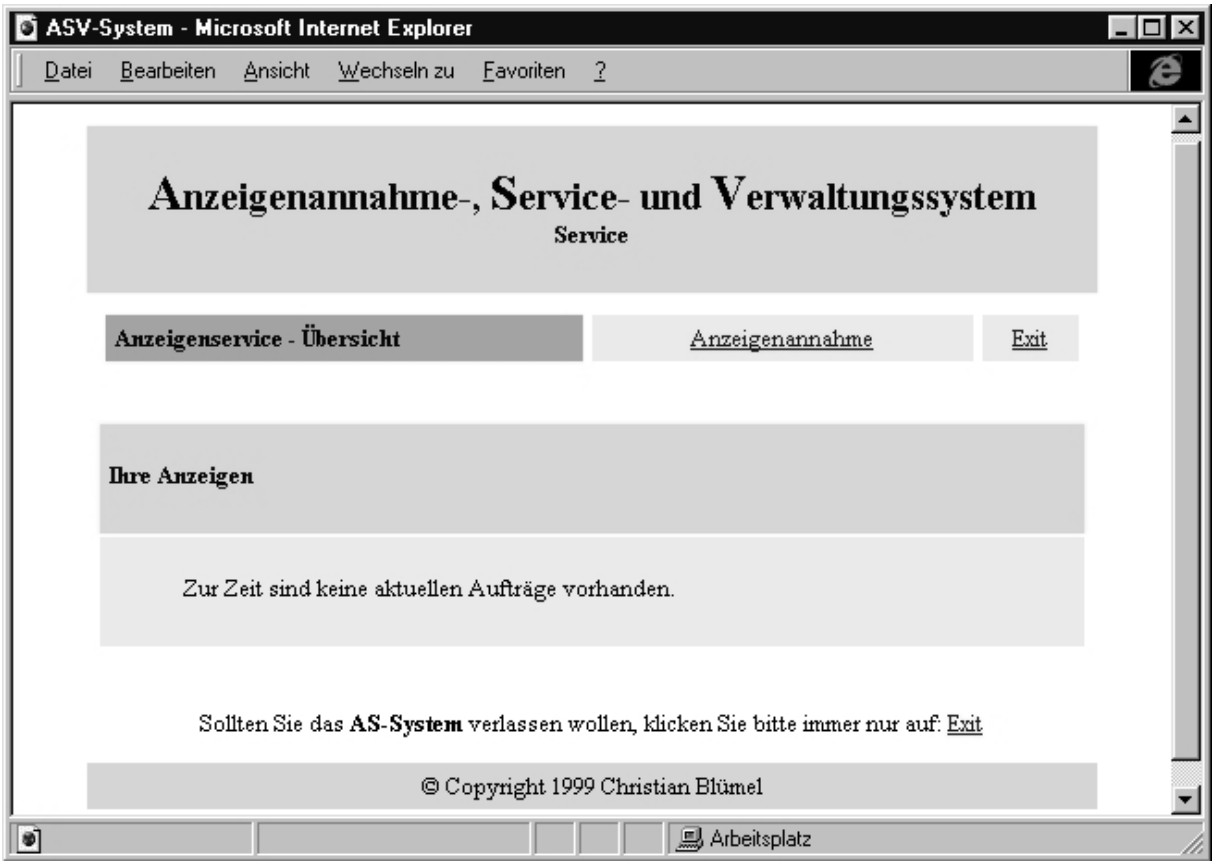

Abb. 12 Die Servicearea – noch ohne aktuelle Aufträge

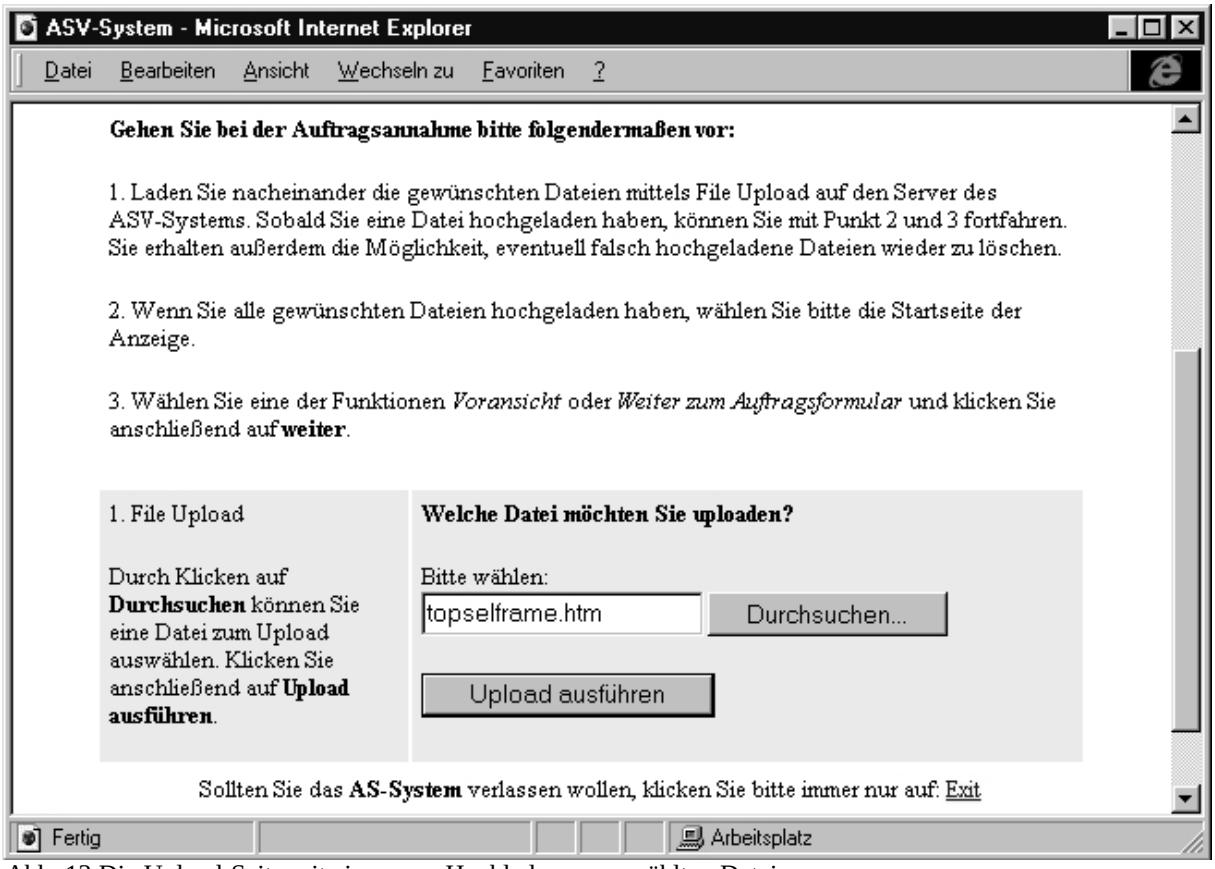

Abb. 13 Die Upload-Seite mit einer zum Hochladen ausgewählten Datei

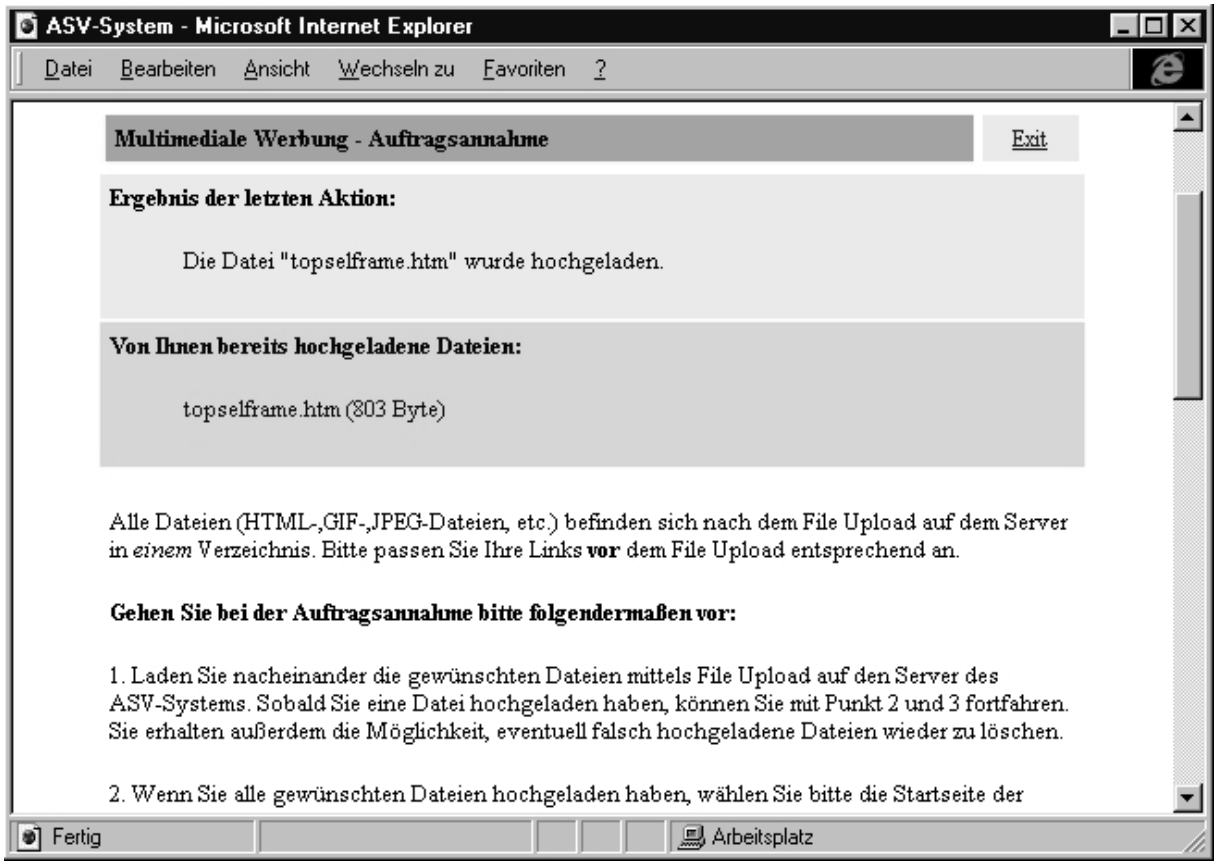

Abb. 14 Bestätigung des erfolgreichen Uploads durch das ASV-System

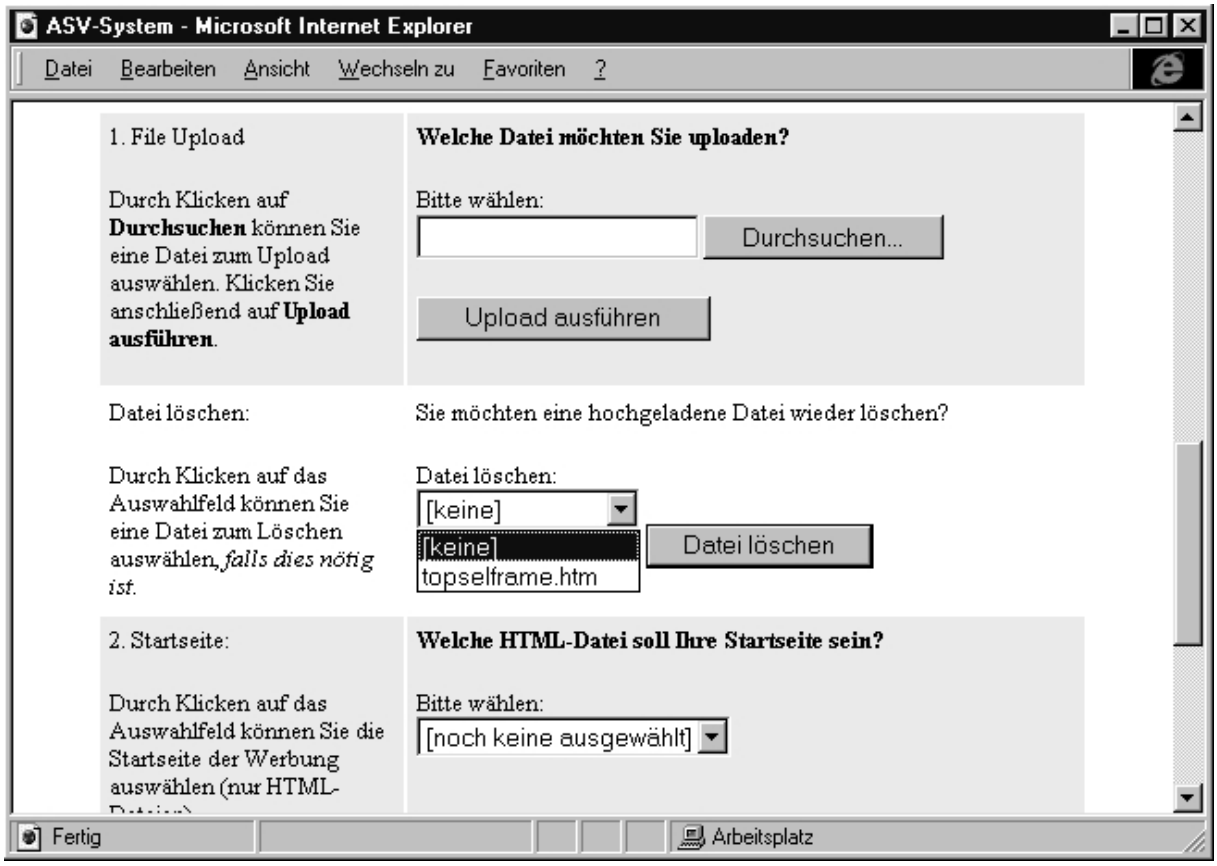

Abb. 15 Möglichkeit zum Löschen einer Datei

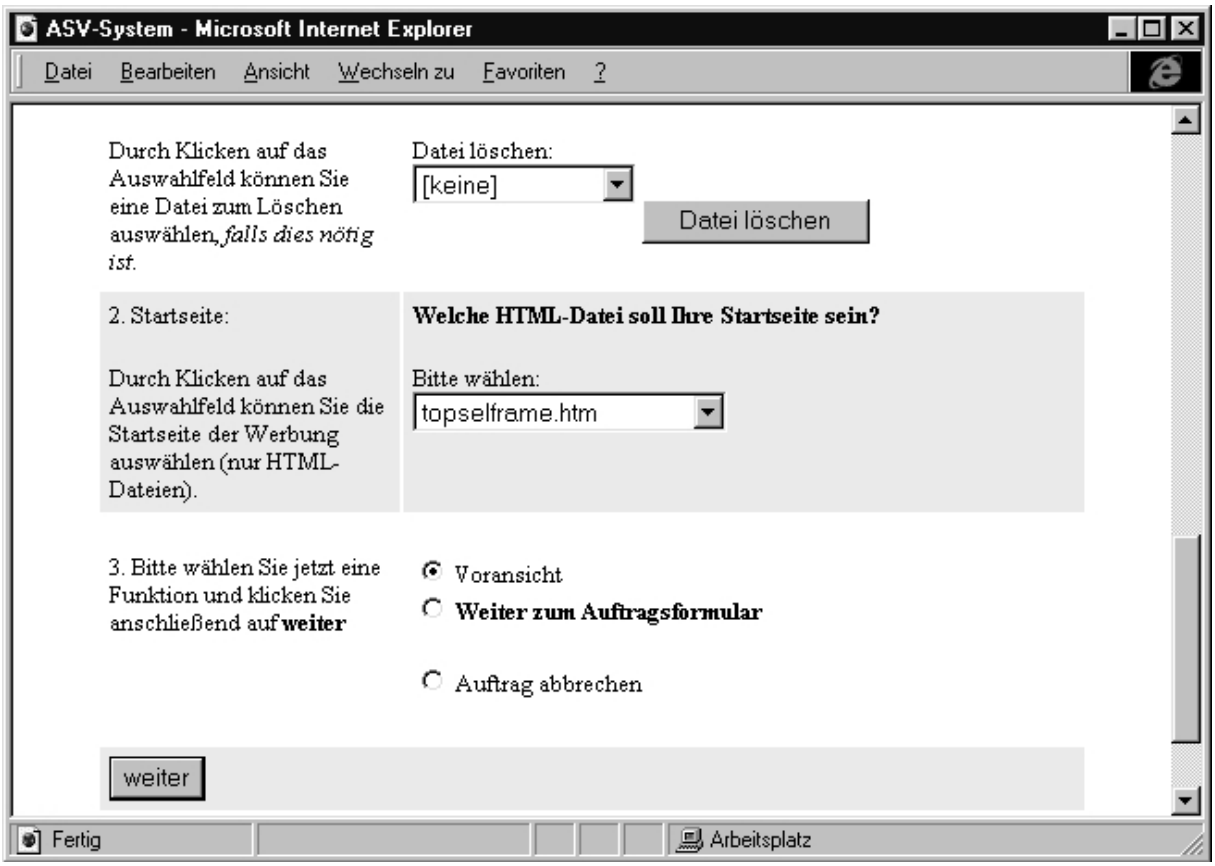

Abb. 16 Auswahl einer Startseite für die Voransicht

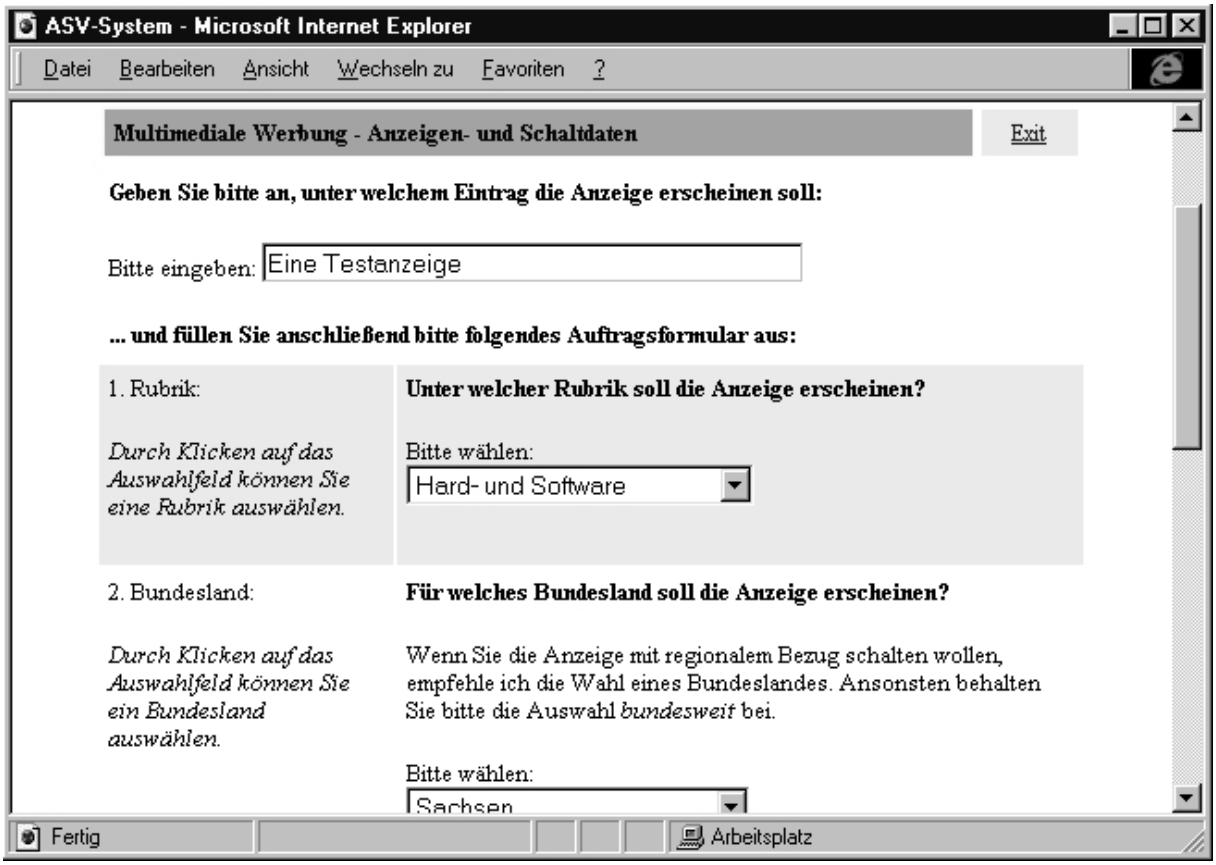

Abb. 17 Eingabe von Anzeigen- und Schaltdaten I

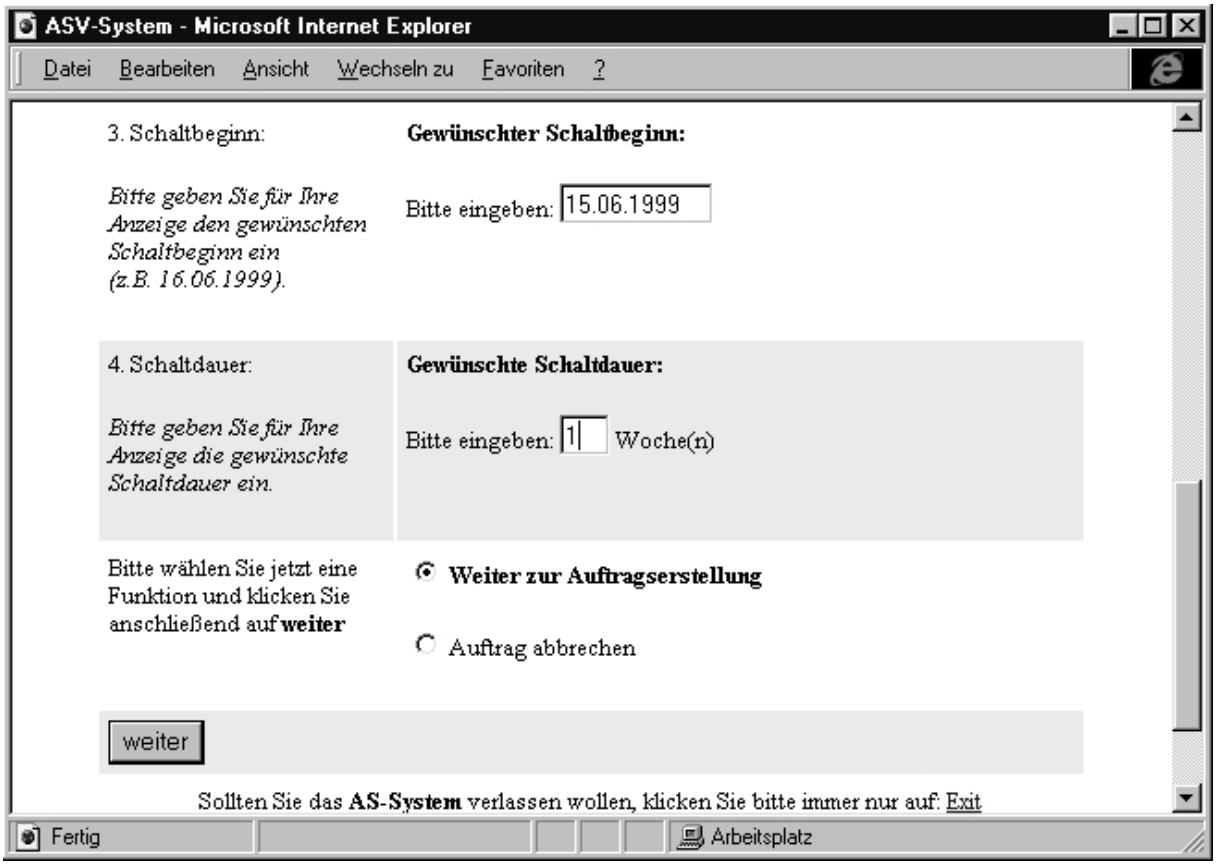

Abb. 18 Eingabe von Anzeigen- und Schaltdaten II

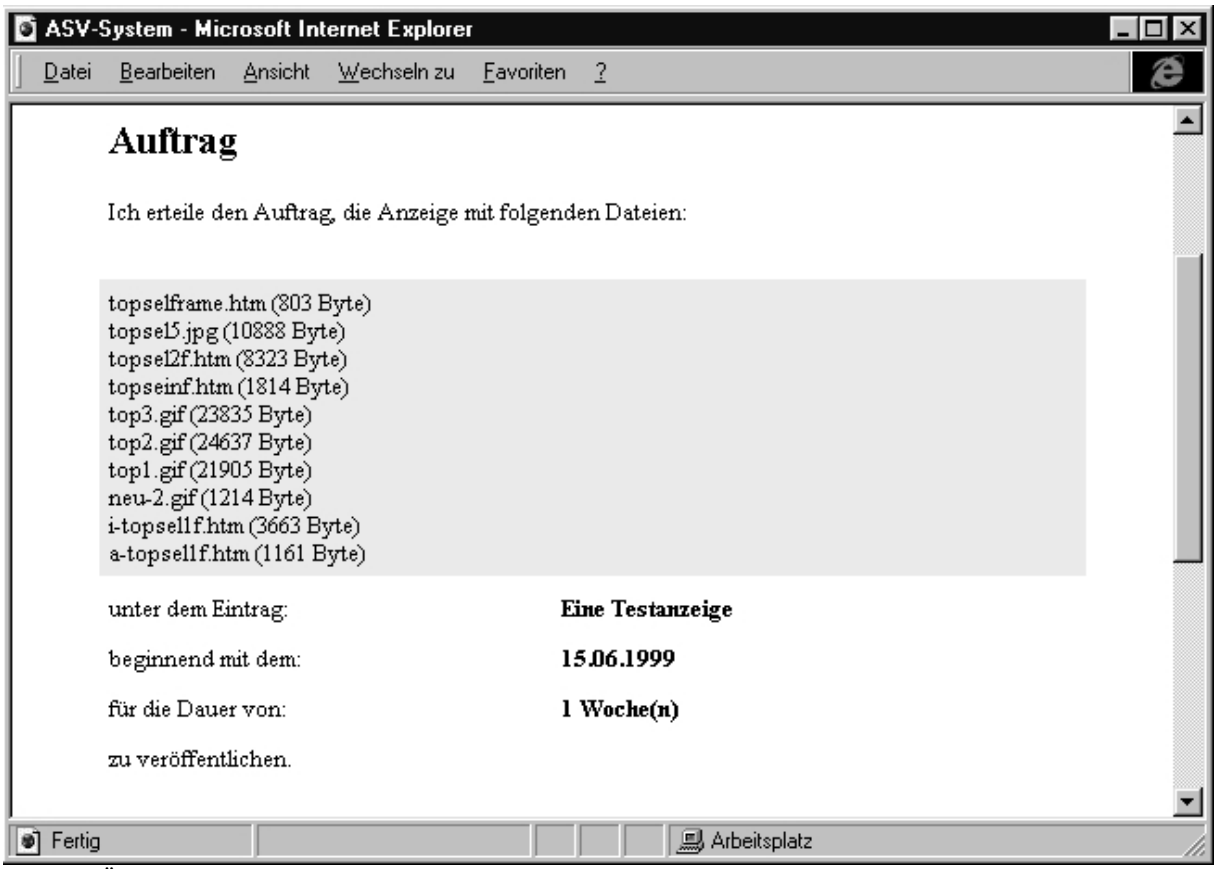

Abb. 19 Übersichtsseite mit Dateinamen sowie Anzeigen- und Schaltdaten

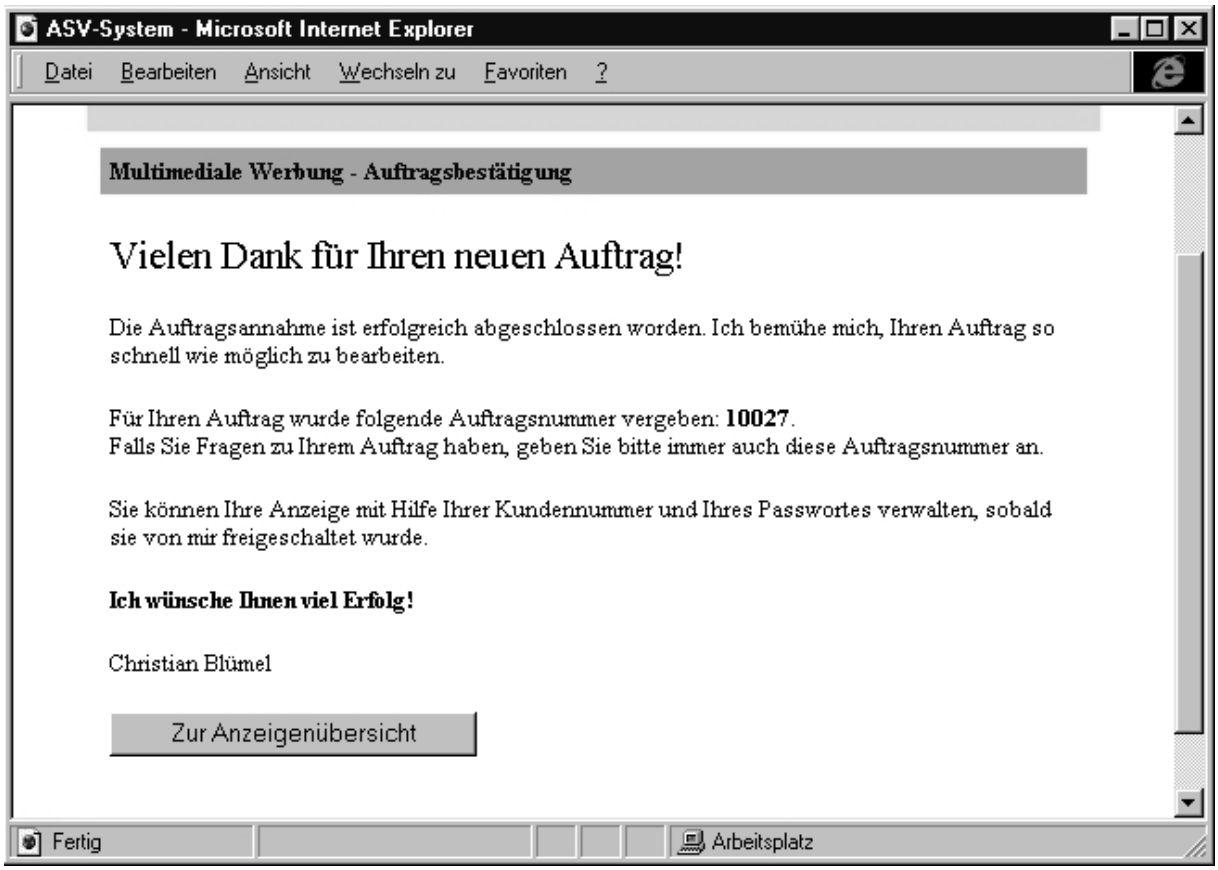

Abb. 20 Bestätigung der erfolgreichen Auftragsannahme

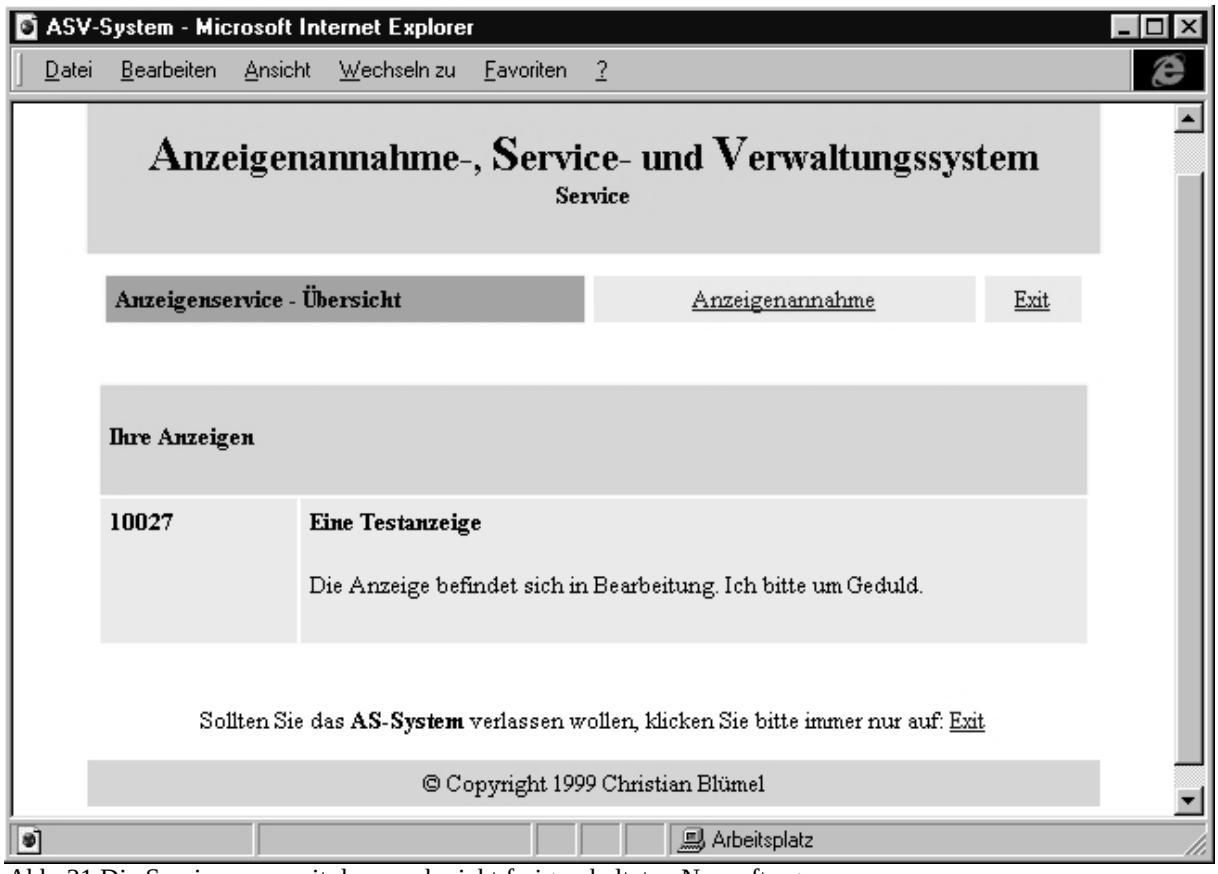

Abb. 21 Die Servicearea - mit dem noch nicht freigeschalteten Neuauftrag

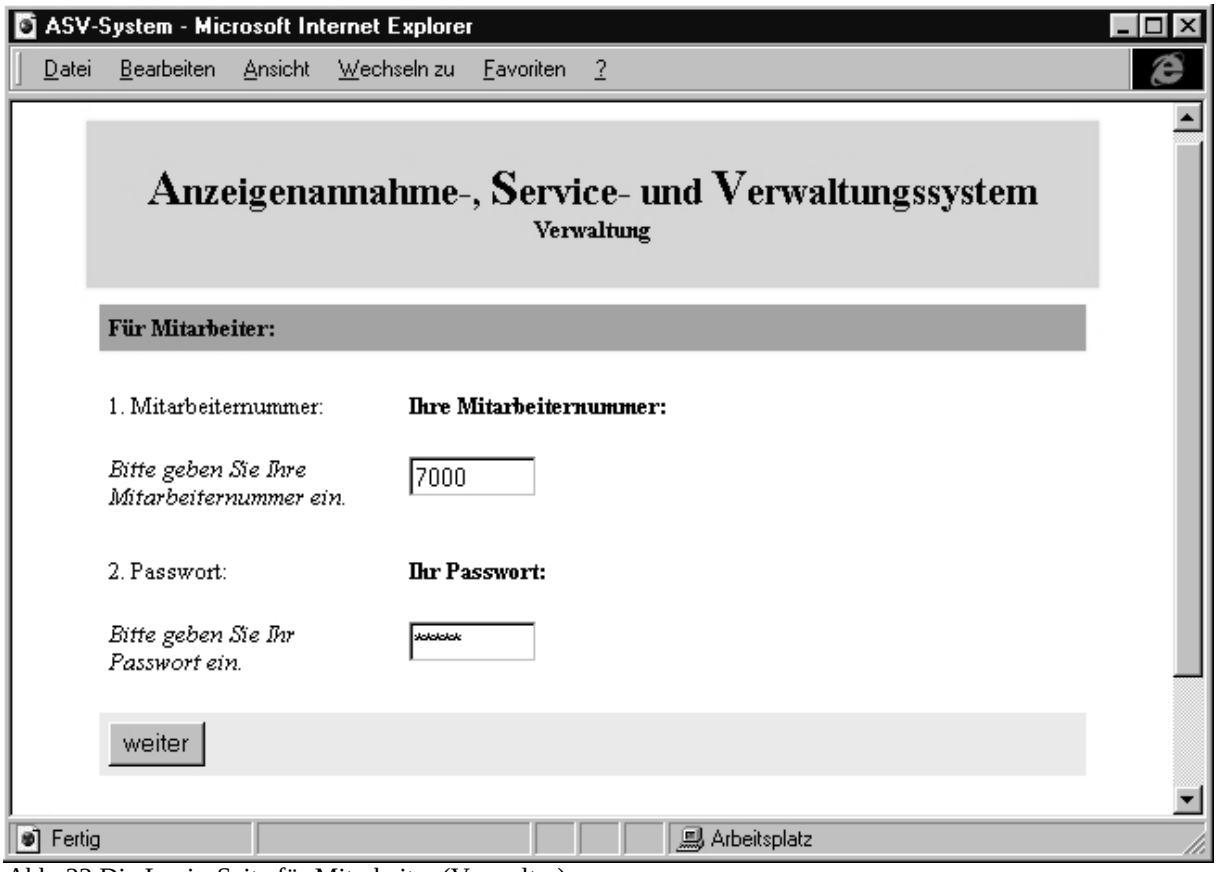

Abb. 22 Die Login-Seite für Mitarbeiter (Verwalter)

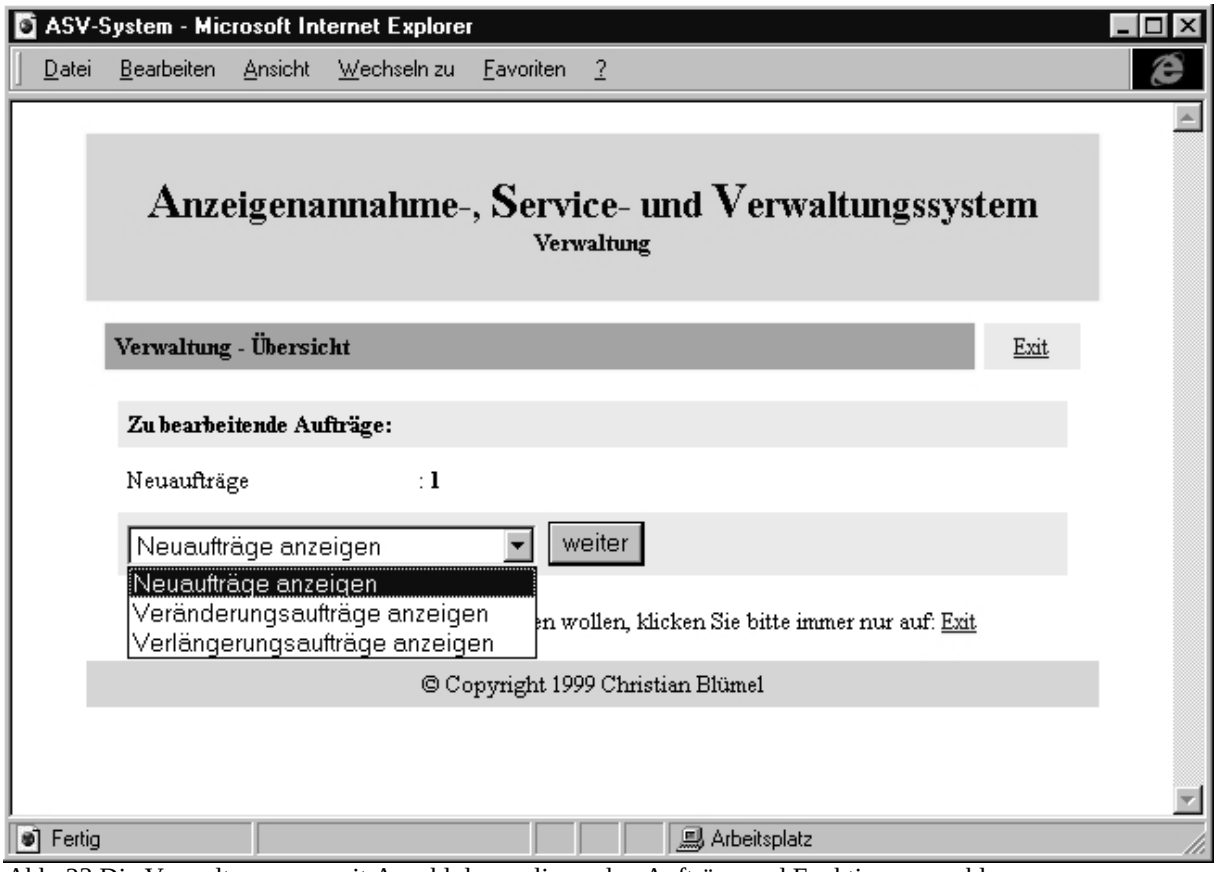

Abb. 23 Die Verwaltungsarea mit Anzahl der vorliegenden Aufträge und Funktionsauswahl

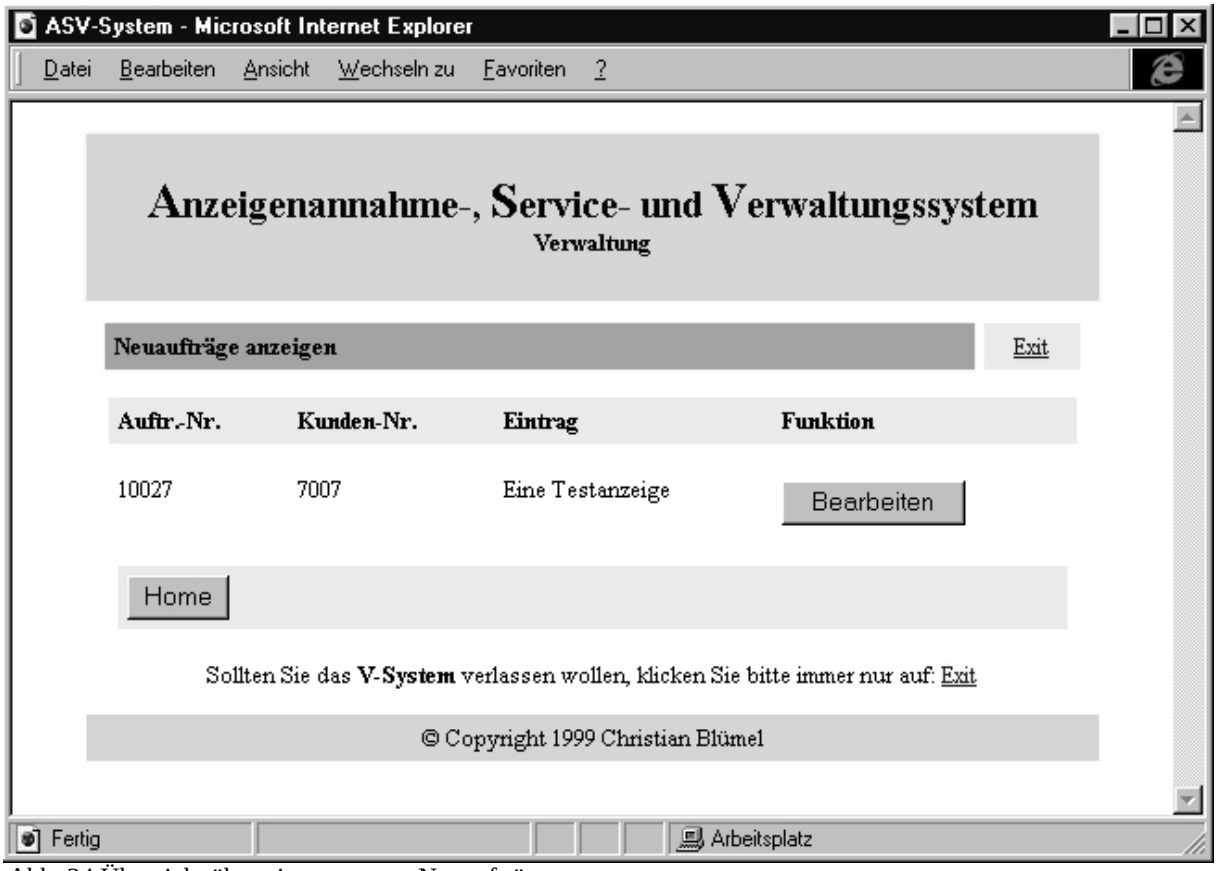

Abb. 24 Übersicht über eingegangene Neuaufträge

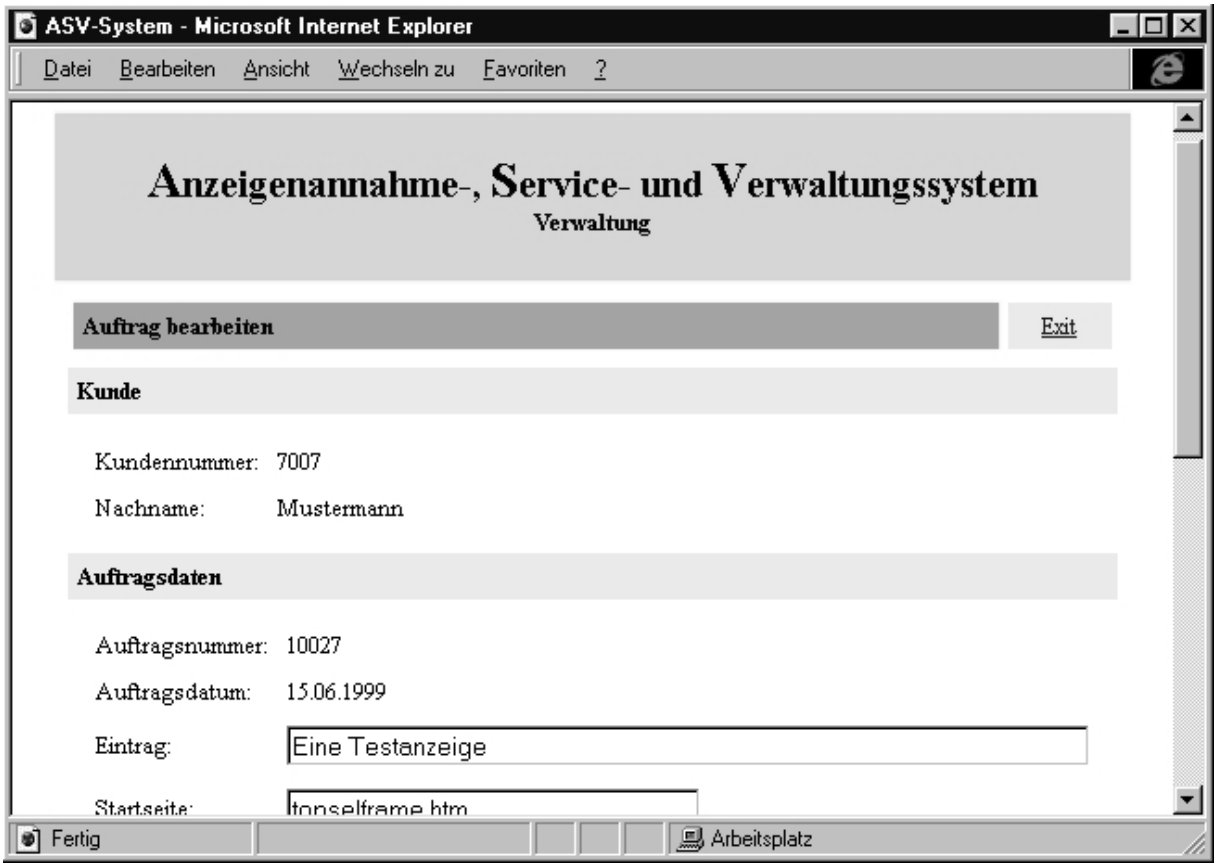

Abb. 25 Detailansicht des Auftrages I

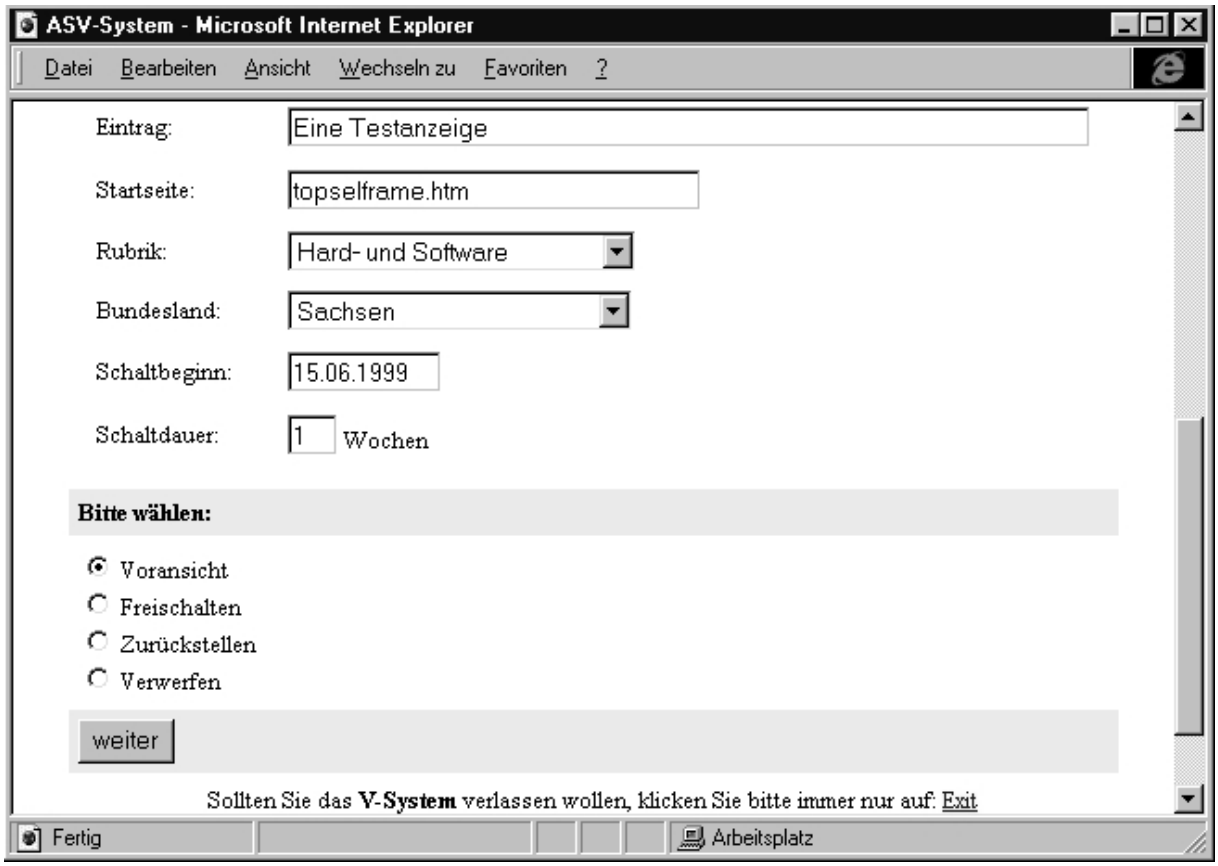

Abb. 26 Detailansicht des Auftrages II

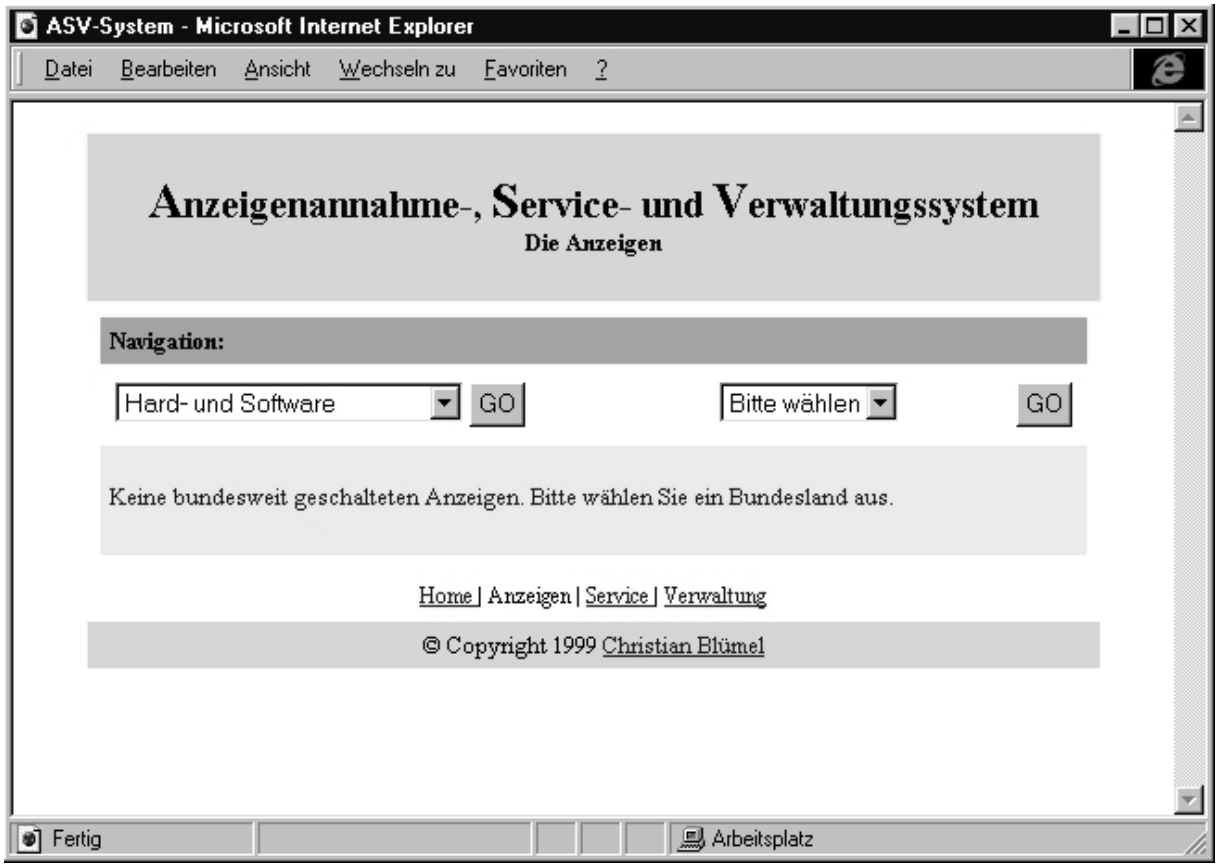

Abb. 27 Nutzerinterface I

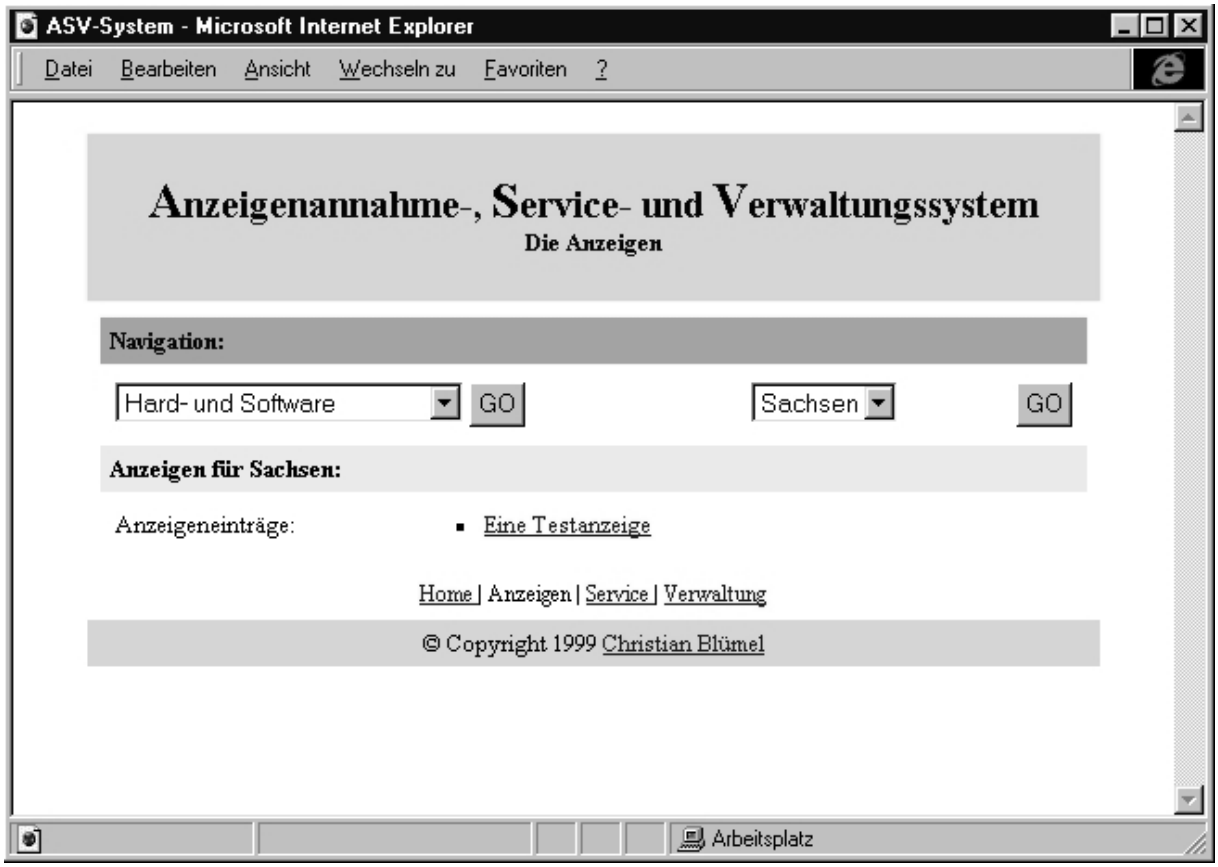

Abb. 28 Nutzerinterface II

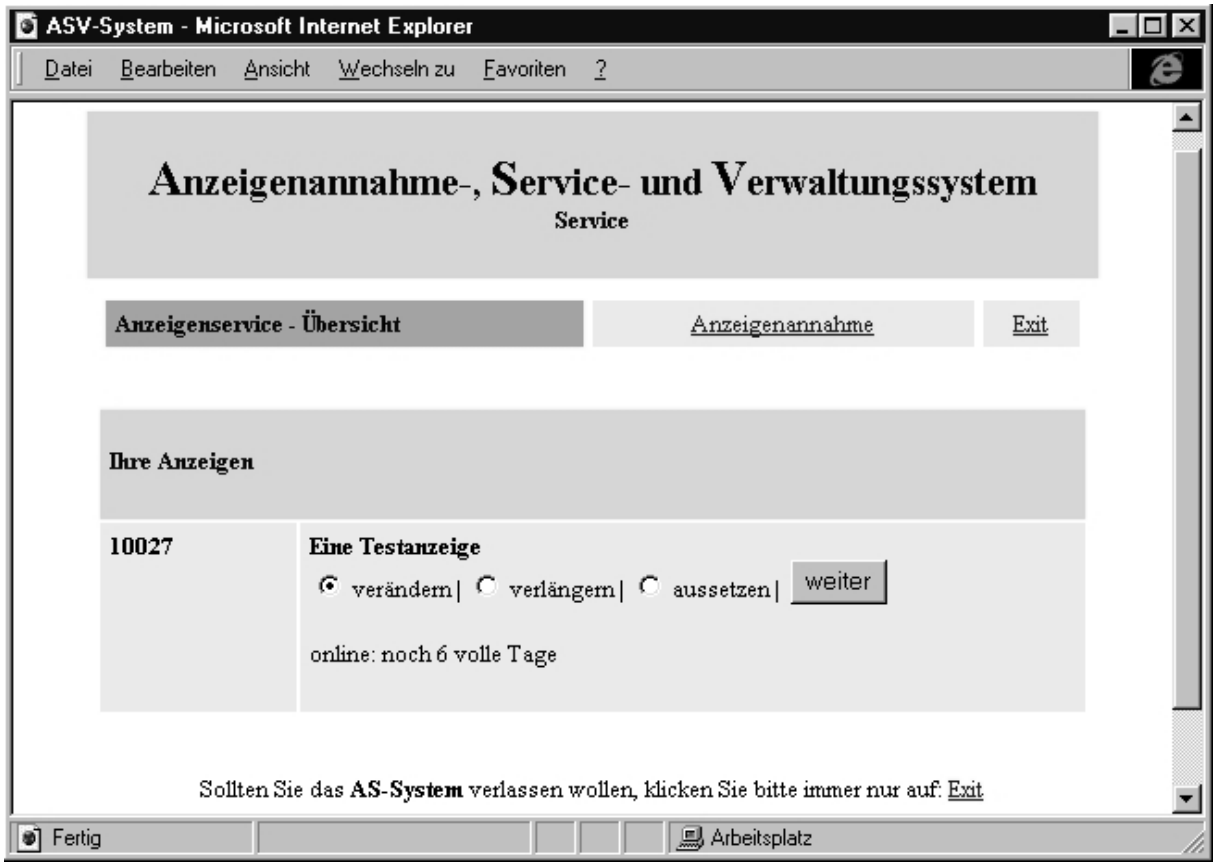

Abb. 29 Servicearea mit Funktionsauswahl für den Auftrag

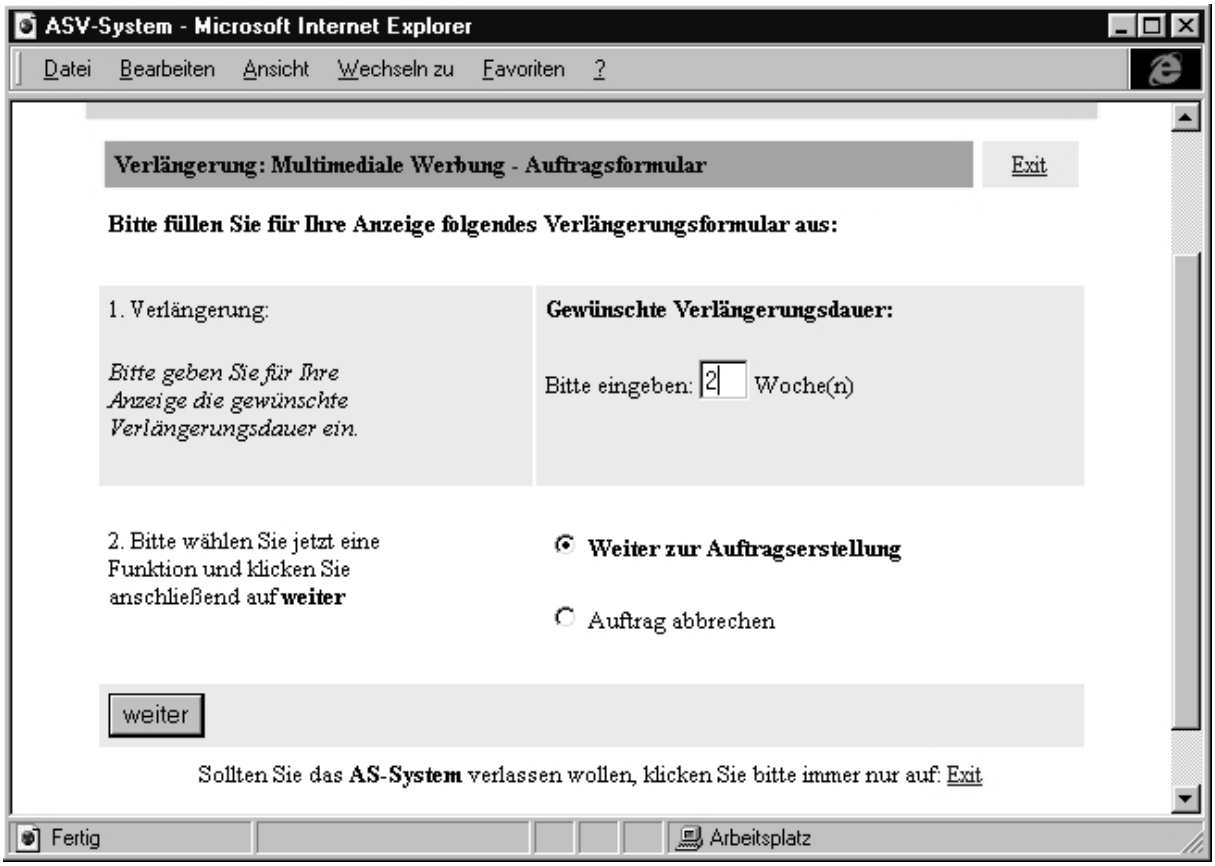

Abb. 30 Verlängerungsformular

III. Diskussion

Die Ziele der Diplomarbeit wurden erreicht. Das ASV-System wurde mittels Verfahren des Software-Engineerings in Planungs-, Definitions- und Entwurfsphase konzipiert und schließlich ein Beispiel-ASV-System auf der Grundlage dieser Konzeption implementiert. Diese Beispielrealisierung ermöglicht es, den gesamten Anzeigenprozeß vom Inserieren der gestalteten Anzeige über das Freischalten in der Verwaltung bis hin zur Ansicht der Anzeige zu demonstrieren. Anzeigen können mit dem Beispielsystem verändert und verlängert sowie von der Ansicht ausgesetzt werden. Der gesamte Anzeigenprozeß kann ausschließlich unter Verwendung eines üblichen WWW-Browsers durchgeführt werden.

Im folgenden sollen die Eignung der verwendeten Methoden des Software-Engineerings für die Erstellung von WWW-Anwendungen und die gefundene technische Lösung für das ASV-System beurteilt werden. Abschließend werden Einsatzmöglichkeiten des ASV-Systems diskutiert.

1. Verwendete Methoden des Software-Engineerings

Die eingesetzte Methode der Strukturierten Analyse (SA) ermöglichte zusammen mit dem Basiskonzept Entity-Relationship-Modell eine schrittweise Annäherung an das Endprodukt. Die dabei entstandenen Ergebnisdokumente (Lastenheft, Pflichtenheft, Produktmodell, Datenbankschema) sind im Anhang A bis D dargestellt.

Der Schritt vom Pflichtenheft zum Datenflußdiagramm mit Data Dictionary und Mini-Spezifikation war sehr zeitintensiv. Die Möglichkeit, Datenflußdiagramme schrittweise zu verfeinern, wurde nicht als hilfreich empfunden, da beinahe jede neue Verfeinerungsstufe aufgrund der zutage tretenden Implikationen ein Neuzeichnen der Datenflüsse bedingte. Nach [BA96, S. 410] ist dies aber üblich. Insbesondere irritierte, daß in Datenflußdiagrammen keine Kontrollflüsse verzeichnet werden.
Andererseits waren die entstandenen Datenflußdiagramme ein idealer Ausgangspunkt für die Erstellung der Mini-Spezifikationen (Pseudocode), die wiederum als Vorlage für die Implementierung dienen.

Kritisch muß bemerkt werden, daß die Basiskonzepte der Strukturierten Analyse kein Framework für die Konzeption von Client-Server-Anwendungen im allgemeinen und WWW-Anwendungen im speziellen bieten. So mußte eine Vielzahl von Themengebieten gesondert – und ohne Unterstützung durch entsprechende Verfahren – erarbeitet werden. Dazu zählten die Auswahl einer geeigneten Produkt- und Entwicklungsumgebung und die Überlegungen zu den Anforderungen an das ASV-System als Client-Server-Anwendung.

Insbesondere die dabei zu konzipierende Sitzungsverwaltung sollte als wichtiger Bestandteil jeder Client-Server-Anwendung Aufnahme in ein geeignetes Framework finden. Auch die Ausarbeitung der relevanten Sicherheitsaspekte in der Entwurfsphase fand ohne Unterstützung durch entsprechende Verfahren statt. Klar ist, daß aufgrund der ständigen Weiterentwicklung konkrete technische

Lösungen keinen Platz in einem zu entwickelnden Framework haben. Interessant wäre aber die Untersuchung, welche Erkenntnisse aus der Entwicklung des ASV-Systems sich in Form eines Frameworks für die Entwicklung von WWW-Anwendungen verallgemeinern lassen.

2. Beurteilung der technischen Lösung

Das Beispiel-ASV-System läuft stabil und erfüllt dabei die in Planungs- und Definitionsphase vereinbarten Qualitätszielbestimmungen.

Durch die Realisierung als CGI-Anwendung und die Verfügbarkeit der Programmiersprache PHP und der Datenbank MySQL sowohl unter Unix als auch unter WindowsNT ist das ASV-System theoretisch nahezu auf allen WWW-Servern einsatzbereit. Erprobt wurde der Einsatz unter Linux mit dem Apache-WWW-Server. Durch die Verwendung einer aus Freeware-Komponenten bestehenden Entwicklungsumgebung kann das ASV-System praktisch von jedem Interessierten installiert, betrieben und weiterentwickelt werden.

Die gewählte Realisierung in Form von HTML-Formularen ermöglicht, daß der Betrieb des Anzeigendienstes ausschließlich unter Verwendung eines WWW-Browsers durchgeführt werden kann und genügt damit den Hauptanforderungen.

Der Einsatz von Java-Script sollte noch einmal überdacht werden, da durch die client-seitige Überprüfung der Eingabedaten vor dem Absenden des Formulars an den Server im Falle von Eingabefehlern der sonst für die Behebung des Fehlers benötigte Datenverkehr entfallen kann. Weiterhin gilt, daß der Einsatz von JAVA-Script nur als *Add-On* verstanden werden kann und eine server-seitige Überprüfung der Eingabedaten in jedem Fall stattfinden muß, da der Nutzer Java-Script deaktiveren kann.

Bei der Implementierung zeigten sich deutlich die Grenzen der gewählten Programmiersprache PHP. So wurden die Quellcodedateien aufgrund der Integration von HTML- und Programmierbefehlen schnell unübersichtlich. Die deshalb vorgenommene Partitionierung in mehrere Quellcodedateien, die bei Bedarf hinzugeladen werden, brachte das Problem der Fehlerlokalisierung mit sich. Die Fehlersuche nach *Syntax Errors* dauert mitunter länger als zehn Minuten, da bisher kein Compiler zur Verfügung steht, der den Ursprungsort eines Fehlers genau lokalisieren kann. So erzeugten fehlende Klammern in einer Quellcodedatei Fehler in einer anderen Quellcodedatei.

Die Integration der Datenbank MySQL verlief dagegen problemlos.

Bei kleineren Projekte spart PHP aufgrund der bereitgestellten, mächtigen WWW- und Datenbankfunktionen und durch die Möglichkeit, den Code ohne Compilieren testen zu können, Zeit. Bei der Programmentwicklung des ASV-Systems wurde dieser Zeitgewinn jedoch aufgrund der obengenannten Nachteile aufgezehrt. Es entstand der Eindruck, daß mit dem Beispiel-ASV-System die Grenze der derzeit mit PHP produktiv implementierbaren Anwendungen erreicht wurde. Allerdings schreitet auch die Entwicklung von PHP voran: Ein *Pre-Compiler* soll sich bereits in der Entwicklung befinden.

3. Einsatzmöglichkeiten des ASV-Systems

Das ASV-System kann als Grundlage für private Anzeigendienste genutzt werden. Im Vergleich zu derzeit verfügbaren Anzeigendiensten bietet das ASV-System für einen Kunden, der gestaltete Anzeigen schalten will, eine sehr viel größere Funktionalität. Nach bisherigem Kenntnisstand gibt es derzeit keinen öffentlich zugänglichen Anzeigendienst, der wie das ASV-System die Möglichkeit bietet, gestaltete Anzeigen in Form von mehreren HTML-Seiten mit GIF-Grafiken, JPEG-Bildern, JAVA-Applets, downloadbaren Dateien, Videos, Sounddateien etc. online zu schalten und während der gesamten Laufzeit online zu verändern.
Eine Verbesserung könnte erzielt werden, indem der Kunde die Möglichkeit erhält, abgelaufene Anzeigen wieder zu aktivieren. Bisher können Anzeigen nur verlängert werden.

Für einen eventuellen kommerziellen Einsatz sind bestimmte Erweiterungen notwendig. Zum einen wurden wichtige Teile einer Anwendung wie eine Kunden- und Mitarbeiterverwaltung, die es ermöglichen sollte - ebenfalls nur unter Nutzung eines WWW-Browsers - neue Kunden und Mitarbeiter anzumelden und dann freizuschalten, bisher noch nicht konzipiert. Die Konzeption und Realisierung einer Kunden- und Mitarbeiterverwaltung kann jedoch orthogonal zum bisherigen ASV-System erfolgen, da die notwendige Schnittstelle in Form der Tabelle *Benutzer* in der Datenbank bereits existiert.

Weiterhin wäre interessant, wie die Realisierung einer Abrechnungsfunktion vorgenommen werden kann. So müßte sowohl eine eigenständige Lösung (Buchhaltung im ASV-System) als auch eine Schnittstelle zu externen Programmen (etwa über den Austausch von formatierten Buchhaltungsdateien) diskutiert werden.

Zum anderen erscheint eine Weiterentwicklung der Sitzungsverwaltung bei einem kommerziellen Einsatz in Hinblick auf die in Abschnitt II.3.2.2 diskutierten Sicherheitsaspekte notwendig.

Beim Einsatz in spezialisierten Anzeigendiensten (Stellen-, Auto-, Immobilienbörsen) müßte zudem die Anzahl der wählbaren Kritierien erweiterbar sein. Bei einer Autobörse wären zum Beispiel *Leistung, Farbe, Sonderausstattungen* etc. sinnvolle Unterscheidungskriterien, die der Nutzer für die Selektion der gewünschten Inserate nutzen könnte. Stellenbörsen dagegen könnten die Einordnung der Inserate nach *Zusatzqualifikationen, Berufserfahrung, Vergütung, Firmenstandort, Einsatzort* etc. vornehmen wollen.
Wünschenswert wäre hier die Möglichkeit, die Konfiguration von Rubriken, Unterrubriken sowie weiteren, frei wählbaren Attributen in der Verwaltung durchführen zu können.

IV. Zusammenfassung

Ausgehend von der Aufgabenstellung, ein Anzeigenannahme-, Service- und Verwaltungssystem (ASV-System) für das Inserieren und Verwalten gestalteter Anzeigen im WWW zu entwickeln, das eine Lösungsmöglichkeit für die Einschränkungen und Problemstellungen aktueller Anzeigen-Online-Dienste im WWW aufzeigt, wurden die Hauptforderungen an das ASV-System formuliert.

Es sollte ein ASV-System entwickelt werden, das es Kunden des Anzeigen-Online-Dienstes ermöglicht, gestaltete Anzeigen in Form von HTML-Seiten mit eingebundenen Grafiken, Bildern, JAVA-Applets, Videos etc. nur unter Verwendung eines üblichen WWW-Browsers zu schalten und während der gesamten Laufzeit der Anzeige verändern zu können.
Die Mitarbeiter (Verwalter) des Anzeigen-Online-Dienstes sollten – ebenfalls nur unter Verwendung eines üblichen WWW-Browsers – die eingegangenen Aufträge bearbeiten und freischalten können.
Die zu entwickelnde Konzeption sollte schließlich in Form eines Beispiel-ASV-Systems realisiert werden.

Die Entwicklung des ASV-Systems wurde nach Verfahren des Software-Engineerings durchgeführt. So wurden schrittweise das Lastenheft, das Pflichtenheft, das Produktmodell und die Konzeption der Benutzungsoberfläche erarbeitet.

Besondere Aufmerksamkeit wurde dabei der Erstellung des Produktmodells gewidmet, das mittels der Methode der Strukturierten Analyse und unter Verwendung des Basiskonzeptes Entity-Relationship-Modell entwickelt wurde. Es besteht aus den Teilen ER-Modell, Datenflußdiagramm, Data Dictionary und einer ausführlichen Mini-Spezifikation in Form von Pseudocode.

Das Datenbankschema des ASV-Systems wurde auf der Grundlage des zuvor erarbeiteten ER-Modells in der Entwurfsphase erstellt.

Die Entwicklung des ASV-Systems als WWW-Anwendung erforderte die Bearbeitung einer Reihe von Problemstellungen, für die die traditionellen Methoden des Software-Engineerings keine adäquate Unterstützung anboten. So mußten Problemstellungen wie die Analyse der Anforderungen an das ASV-System als Client/Server-Anwendung, die Auswahl der geeigneten Entwicklungswerkzeuge und die Analyse der relevanten Sicherheitsaspekte gesondert durchgeführt werden.

Das auf der Grundlage der erstellten Konzeption realisierte Beispiel-ASV-System erfüllt die gestellten Hauptanforderungen und könnte – bestimmte Erweiterungen wie zum Beispiel eine Benutzerverwaltung vorausgesetzt – als Grundlage für die Realisierung einer Vielzahl von Online-Diensten (Anzeigendienste, Stellen- und Immobilienbörsen etc.) genutzt werden.

Literaturverzeichnis

[BA96] Balzert, H.: *Lehrbuch der Software-Technik*, 1. Auflage, Spektrum Akademischer Verlag Heidelberg; Berlin; Oxford, 1996

[CB96] Cheswick, W. R.; Bellovin, S. M.: *Firewalls und Sicherheit im Internet*, 1. Auflage, Addison-Wesley, Bonn; Paris; New York, 1996

[EI98] Eilebrecht, L.: *Apache Web Server*, 2. Auflage, ITP Verlag GmbH, Bonn, 1998

[KU98] Kunze, M.: *LAMP – Datenbankgestütztes Web-Publishing-System mit Freeware,* In: c't Magazin für Computer Technik, Heinz Heise Verlag, S. 230-231, 12/1998

[LO98] Loeser, H.: *Techniken für Web-basierte Datenbankanwendungen: Anforderungen , Ansätze, Architekturen,* In: Informatik Forschung und Entwicklung, Springer-Verlag, S.196-216, 13/1998

[MA96] Maurer, R.: *HTML und CGI-Programmierung: Dynamische WWW-Seiten erstellen,* 1. Auflage, dpunkt, Verlag für Digitale Technologie Heidelberg, S.135ff., 1996

[NF96] Niederst, J.; Freedman, E.: *Web-Design: Gestaltungsgrundlagen für ein neues Medium,* 1. Auflage, O´Reilly International Thomson Verlag Bonn, S.80, 1996

[OH99] Orfali, R.; Harkey, D.; Edwards, J.: *The Essential Client/Server Survival Guide*, 3. Auflage, Wiley Computer Publishing New York; Toronto u.a., S. 192, 1999

[RA97] Rahm, E.: *Datenbanksysteme I*, Skript zur Vorlesung, Wintersemester 1997/1998

[SA94] Sauer, H.: *Relationale Datenbanken: Theorie und Praxis*, 3., korr. Auflage, Addison-Wesley Bonn; München; Paris, S. 13, 1994

[TG96] Tittel, E.; Gaither, M.; Hassinger, S.; Erwin, M.: *WWW-Programmierung mit CGI: Entwicklung interaktiver Anwendungen,* 1. Auflage, International Thomson Publishing Bonn; Albany, 1996

Anhang A – Lastenheft

1. Zielbestimmung

Das zu entwickelnde Anzeigenannahme-, Service- und Verwaltungssystem (ASV-System) soll das Publizieren und Verwalten von gestalteten Inseraten im World Wide Web (WWW) ermöglichen.

2. Produkteinsatz

Das ASV-System soll als Grundlage für die Entwicklung von Anzeigendiensten dienen, die Kunden das Inserieren gestalteter Anzeigen im WWW ermöglichen. Während das Anzeigenannahme- und Service-Teilsystem (AS-System) den Kunden der Anzeigendienste als einer Zielgruppe die Schaltung der Inserate und die Kontrolle über die Anzeige während der Laufzeit des Inserates ermöglichen soll, soll das Verwaltungsteilsystem (V-System) dem Betreiber des Dienstes als zweiter Zielgruppe die Verwaltung der Anzeigen gestatten. Sowohl Anzeigenannahme und Service als auch die Verwaltung sollen komplett über das WWW und nur unter Verwendung eines üblichen WWW-Browsers möglich sein. Nutzer der Anzeigendienste sollen sich den Anzeigenbestand nach verschiedenen Kriterien geordnet anzeigen lassen können.

3. Produktfunktionen

/LF 10/ Aufgabe neuer Anzeigen mit Dateitransfer, Möglichkeit zum Dateilöschen, Voransicht, Angabe von Anzeigendaten durch den Kunden

/LF 20/ Freischalten, Löschen, Zurückstellen neuer Anzeigen durch Mitarbeiter

/LF 30/ Änderung von Anzeigeninhalten und Anzeigendaten durch den Kunden

/LF 40/ Freischalten, Löschen, Zurückstellen von durch den Kunden initiierten Änderungen von Anzeigeninhalten und Anzeigendaten

/LF 50/ Darstellung der Anzeigen nach den Kriterien Rubrik oder Kundenname, Suche nach Schlagwörtern

4. Produktdaten

/LD 10/ Es sind folgende Daten zur Verwaltung der Anzeige zu speichern: Überschrift, Schlagwörter, Kurzbeschreibung, Kontakt-Email-Adresse, Rubrik, Unterrubrik, Stadt, Schaltbeginn, Schaltende, Start-HTML-Seite

5. Produktleistungen

/LL 10/ Keine Beschränkung der Anzeigenanzahl durch das ASV-System

/LL 20/ Keine Beschränkung der Anzahl der Dateien pro Anzeige durch das ASV-System, Beschränkung durch Betreiber durch Vorgabe einer max. Anzeigengröße möglich

6. Qualitätsanforderungen

Das System soll stabil gegenüber Fehleingaben sein.

7. Ergänzungen

Die Benutzungsschnittstelle soll den Kunden durch die einzelnen Etappen der Anzeigenannahme (Dateitransfer, Eingabe der Anzeigendaten etc.) und des Anzeigen-Services führen und dabei eine Erfolgskontrolle und Korrektur von einmal gemachten Eingaben ermöglichen.

Anhang B – Pflichtenheft

1. Zielbestimmung

Das zu entwickelnde Anzeigenannahme-, Service- und Verwaltungssystem (ASV-System) soll das Publizieren und Verwalten von gestalteten Inseraten im World Wide Web (WWW) ermöglichen.

1.1 Mußkriterien

- Annahmemöglichkeit von gestalteten Anzeigen in Form von HTML-Seiten mit GIF-Grafiken, JPEG-Bilder, Java-Applets etc. und per Link downloadbaren Programmdateien für Kunden des ASV-Systems, nur unter Verwendung eines WWW-Browsers der Firmen Microsoft ab Version 4.0 und Netscape ab Version 2.0
- Änderungsmöglichkeit der geschalteten Anzeigen für Kunden des ASV-Systems - nur unter Verwendung eines WWW-Browsers
- Freischaltung von neu geschalteten Anzeigen und erteilten Änderungsaufträgen durch den Verwalter des ASV-Systems - nur unter Verwendung eines WWW-Browsers
- Ansicht der geschalteten Anzeigen für die Nutzer, geordnet nach Rubriken oder Kundennamen, Suchmöglichkeit im Anzeigenbestand mittels Schlagwörtern - nur unter Verwendung eines WWW-Browsers

1.2 Wunschkriterien

- Verlängerung der Schaltdauer von Anzeigen durch den Kunden - nur unter Verwendung eines WWW-Browsers

1.3 Abgrenzungskriterien

- Keine integrierte Benutzerverwaltung. Für die Demonstration des ASV-Systems wird ein Kunden- und Mitarbeiteraccount angelegt. Nutzer erhalten freien Zugang zu den Anzeigen
- Keine integrierte Buchführung

2. Produkteinsatz

Das ASV-System soll als Grundlage für die Entwicklung von Anzeigendiensten dienen, die Kunden das Inserieren gestalteter Anzeigen im WWW ermöglichen.

2.1 Anwendungsbereiche

Das ASV-System kann als Grundlage für „Schwarze Bretter“ (Verkaufe – Suche), Stellen-, Wohnungs-, Automärkte oder Ausschreibungs- und Bekanntmachungs-dienste im WWW genutzt werden.

2.2 Zielgruppen

Die Zielgruppe des ASV-Systems sind Diensteentwickler und -betreiber. Nach einer Anpassung an die jeweiligen Bedürfnisse des Betreibers werden Privatpersonen und Unternehmen als Kunden oder Nutzer zusätzlich Zielgruppen des ASV-Systems.

2.3 Betriebsbedingungen

- Büro- und Heimumgebung

3. Produktumgebung

Der Einsatz soll sowohl auf einem gehosteten WWW-Server (Rechner gehört dem Betreiber des Anzeigendienstes, Provider stellt Verbindung zum Internet bereit) als auch auf einer virtuellen Domain (Provider stellt anteilig Plattenplatz auf einem seiner WWW-Server bereit) erfolgen können. Der Einsatz auf einem gehosteten WWW-Server entspricht dabei weitgehend dem Einsatz im Unternehmen des Betreibers des Anzeigendienstes, wenn dieser einen eigenen Rechner mit Stand-leitung ins Internet bereitstellt.

3.1 Software

- Unix-Betriebssystem
- Apache-Webserver ab Version 1.1
- MySQL-Datenbank ab Version 3.22

3.2 Hardware

- Rechner, auf dem ein Unix-Betriebssystem läuft

3.3 Orgware

- Der Zugang zum Internetrechner des Betreibers muß über die Internetdienste FTP und TELNET für die Installation und für die Wartung des ASV-Systems möglich sein
- Auf dem Rechner müssen die Quellcodedateien von Apache, PHP und MySQL mittels GNU C++ kompilierbar sein

4. Produktfunktionen

4.1 Anzeigenannahme

/F10/ Einloggen mit Kundennummer und Passwort
/F20/ Upload von Dateien /LF10/ mit
/F30/ - Anzeige der hochgeladenen Dateien mit Dateiname und Dateigröße
/F35/ - Überprüfung, ob die definierte Obergrenze von x MB nicht überschritten wurde (wenn ja -> Löschen der zuletzt hochgeladenen Datei)
/F40/ - Möglichkeit zum Löschen hochgeladener Dateien mit
/F50/ - Auswahlmöglichkeit unter den bereits hochgeladenen Dateien
/F60/ - Möglichkeit zur Auswahl einer Start-HTML-Seite mit
/F70/ - Auswahlmöglichkeit unter den bereits hochgeladenen HTML-Dateien
/F80/ - Möglichkeit zur Voransicht der Anzeige
/F90/ - Erfassung von Anzeigen- und Schaltdaten
/F100/ Ausgabe einer Übersicht vor der Auftragsbestätigung mit allen Anzeigendateien, Anzeigen- und Schaltdaten
/F110/ Ausgabe einer Bestätigungsseite bei erfolgreich abgeschlossener Annahme

4.2 Anzeigenservice

/F120/ Einloggen in das Servicesystem mit Kundennummer und Passwort
/F130/ Ausgabe einer Übersicht aller bisher geschalteten Anzeigen mit Funktionsauswahl je Anzeige (*Verändern*, *Verlängern*, *Aussetzen)* und Ausgabe der Restlaufzeit je Anzeige
/F140/ Verändern einer Anzeige /LF 30/ mit
/F150/ - Upload von neuen Dateien /F20-F80/, wobei die
/F160/ - bereits vorhandenen Dateien angezeigt werden
/F170/ - Ändern von Anzeigen- und Schaltdaten /F90/, wobei die
/F180/ - bereits vorhandenen Daten angezeigt werden
/F190/ - Ausgabe einer Übersicht vor der Auftragsbestätigung mit allen Anzeigendateien, Anzeigen- und Schaltdaten
/F200/ - Ausgabe einer Bestätigungsseite bei erfolgreich abgeschlossener Veränderung
/F203W/ Verlängern einer Anzeige
/F205/ Aussetzen einer Anzeige

4.3 Anzeigenverwaltung

/F210/ Einloggen in das Verwaltungssystem mit Mitarbeiternummer und Passwort
/F220/ Ausgabe einer Übersicht mit der Anzahl der neu eingegangenen Anzeigen, der Anzahl der Veränderungs- und Verlängerungsaufträge und der Anzahl der bisher geschalteten Anzeigen sowie der Anzahl der derzeit aktiven Anzeigen
/F230/ Ausgabe einer Funktionsauswahl (*Neue Anzeigen freischalten*, *Veränderungen freischalten*, *Verlängerungen freischalten)*
/F240/ Freischalten neuer Anzeigen /LF20/ mit

/F250/ - Übersicht aller eingegangenen Anzeigen mit Anzeigen- und Kundendaten und Funktionsauswahl je Anzeige (*Anzeige sofort freischalten*, *Anzeige bearbeiten*, *Anzeige verwerfen)*
/F260/ - Anzeigen sofort freischalten
/F270/ - Anzeige bearbeiten mit
/F280/ - Ansicht/Bearbeitung hochgeladener Dateien, Anzeigen- und Schaltdaten
/F290/ - Möglichkeit, Anzeige endgültig freizuschalten
/F300/ - Möglichkeit, Anzeige zurückzustellen (vorgenommene Änderungen des Verwalters werden rückgängig gemacht)
/F310/ - Möglichkeit, Anzeige zu verwerfen
/F320/ - Anzeige verwerfen /F310/
/F330/ Freischalten von Veränderungsaufträgen /LF 40/ mit
/F340/ - Übersicht aller eingegangenen Veränderungsaufträge mit Anzeigen- und Kundendaten und Funktionsauswahl je Anzeige (*Veränderung sofort freischalten*, *Veränderung bearbeiten*, *Veränderung verwerfen)*
/F350/ - Veränderungen sofort freischalten
/F360/ - Veränderungen bearbeiten mit
/F370/ - Ansicht/Bearbeitung hochgeladener Dateien, Anzeigen- und Schaltdaten
/F380/ - Möglichkeit, Veränderung endgültig freizuschalten
/F390/ - Möglichkeit, Veränderung zurückzustellen (vorgenommene Änderungen des Verwalters werden rückgängig gemacht)
/F400/ - Möglichkeit, Veränderung zu verwerfen
/F410/ - Veränderung verwerfen /F400/
/F420W/ Freischalten von Verlängerungsaufträgen mit
/F430W/ - Übersicht aller eingegangenen Verlängerungssaufträge mit Anzeigen- und Kundendaten und Funktionsauswahl je Anzeige (*Verlängerung sofort freischalten*, *Verlängerung bearbeiten*, *Verlängerung verwerfen)*
/F440W/ - Verlängerungsauftrag sofort freischalten
/F450W/ - Verlängerungsauftrag bearbeiten mit
/F460W/ - Ansicht/Bearbeitung der Verlängerungsdauer
/F470W/ - Möglichkeit, Verlängerungsauftrag endgültig freizuschalten
/F480W/ - Möglichkeit, Verlängerungsauftrag zurückzustellen (vorgenommene Änderungen des Verwalters werden rückgängig gemacht)
/F490W/ - Möglichkeit, Verlängerungsauftrag zu verwerfen
/F500W/ - Verlängerungsauftrag verwerfen /F490W/

4.4 Anzeigendarstellung

/F510/ Darstellung des Anzeigenbestandes /LF 50/ mit
/F520/ - Wahlmöglichkeit für die Strukturierung nach Rubriken oder Kundenname, jeweils geordnet nach Städten und dort wiederum nach Schaltdatum
/F530/ - Suche nach Schlagwörtern der Anzeige
/F540/ Anzeige eines nach /F520-F530/ ausgewählten Inserates am Bildschirm

5. Produktdaten

5.1 Anzeigendaten

/D10/ Über eine Anzeige sind folgende Daten zu speichern: /LD 10/ Anzeigen-Nr., Überschrift, Schlagwörter, Kurzbeschreibung, Kontakt-Email-Adresse, Rubrik, Unterrubrik, geschaltet_für_Städte, Schaltbeginn, Schaltende, Start-HTML-Seite

5.2 Kunden- und Mitarbeiterdaten

/D20/ Über einen Kunden sind folgende Daten zu speichern: Kunden-Nr., Name, Passwort

/D30/ Über einen Mitarbeiter (Verwalter) sind folgende Daten zu speichern: Mitarbeiter-Nr., Name, Passwort

6. Produktleistungen

/L10/ analog /LL 10/

/L20/ analog /LL 20/

/L30/ Zur gleichen Zeit soll eine Anzeige immer nur von einer Person bearbeitet werden können (Wahrung der Integrität).

7. Benutzungsoberfläche

/B10/ Die Benutzungsoberfläche besteht aus HTML-Formularen, die in einem WWW-Browser der Firma Netscape ab Version 2.0 und einem WWW-Browser der Firma Microsoft ab Version 4.0 angezeigt werden.

/B20/ Ein Bearbeitungsprozeß (Anzeigenannahme, Anzeigenservice, Anzeigenverwaltung) besteht jeweils aus der Abarbeitung aufeinanderfolgender HTML-Formulare durch den Kunden oder Verwalter, wobei die Navigationsmöglichkeit zwischen den Formularen sichergestellt werden soll.

/B30/ Der Kunde soll durch in die HTML-Formulare integrierte Hilfe-Informationen durch die Prozesse Anzeigenannahme und Anzeigenservice geleitet werden.

8. Qualitätszielbestimmung

/Q10/ Das System soll stabil gegenüber Fehleingaben des Kunden bei der Eingabe von Anzeigen- und Schaltdaten sein.

/Q20/ Das System soll stabil gegenüber nicht ordnungsgemäßem Abbruch von Bearbeitungsaufträgen (in Anzeigenannahme, -service, -verwaltung) sein

9. Globale Testszenarien/Testfälle

/T10/ Aufgabe einer gestalteten Anzeige bestehend aus mehreren verlinkten HTML-Seiten mit eingebetteten GIF-Grafiken oder JPEG-Bildern
/T20/ Freischalten dieser Anzeige in der Anzeigenverwaltung
/T30/ Vornahme einer Änderung an dieser Anzeige
/T40/ Freischaltung der Änderung
/T50W/ Verlängerung der Schaltdauer einer Anzeige
/T60W/ Freischaltung der Verlängerung

10. Entwicklungsumgebung

10.1 Software

- Linux, Kernel-Version 2.0.3x
- Apache-Webserver, Version 1.1
- MySQL-Datenbank, Version 3.22

10.2 Hardware

- Intel PC

10.3 Orgware

- siehe 3.3 (Orgware für Produktumgebung)

11. Ergänzungen

11.1 Definitionen

Auftrag (Anzeigen-, Änderungs- oder Verlängerungsauftrag):

Vom Kunden vollständig abgeschlossene und vom ASV-System bestätigte Anzeigenannahme, Änderung einer Anzeige oder Verlängerung einer Anzeige

Benutzer

Kunde, Nutzer oder Verwalter (Mitarbeiter)

Kunde

Privatperson oder ein Unternehmen, für die oder das ein Account für das ASV-System eingerichtet wurde, um gestaltete Anzeigen zu schalten

Nutzer

Internetsurfer, der sich die geschalteten Anzeigen betrachten möchte

Restlaufzeit

Zeit, die die Anzeige noch für Nutzer sichtbar ist

Upload (auch „Hochladen“) von Dateien

Übermittlung von Dateien vom Rechner des Kunden an den Rechner des ASV-Systems und Speicherung der Dateien auf dem Rechner des ASV-Systems

Verwalter oder Mitarbeiter

Angestellter oder Betreiber eines Anzeigen-Online-Dienstes, besitzt einen Verwaltungsaccount für das ASV-System, um die Anzeigen zu verwalten (Freischalten neuer Anzeigen, Freischalten von Veränderungsaufträgen, usw.)

WWW-Server (Hardware)

Rechner mit Anschluß an das Internet, auf dem als Software ein WWW-Server (Software, siehe dort) läuft

WWW-Server (Software)

Software, die auf Anfrage eines Nutzers Dokumente, die auf dem WWW-Server (Hardware, siehe dort) gespeichert sind, an den Nutzer liefert

Anhang C – Produktmodell

1. Entity-Relationship-Modell

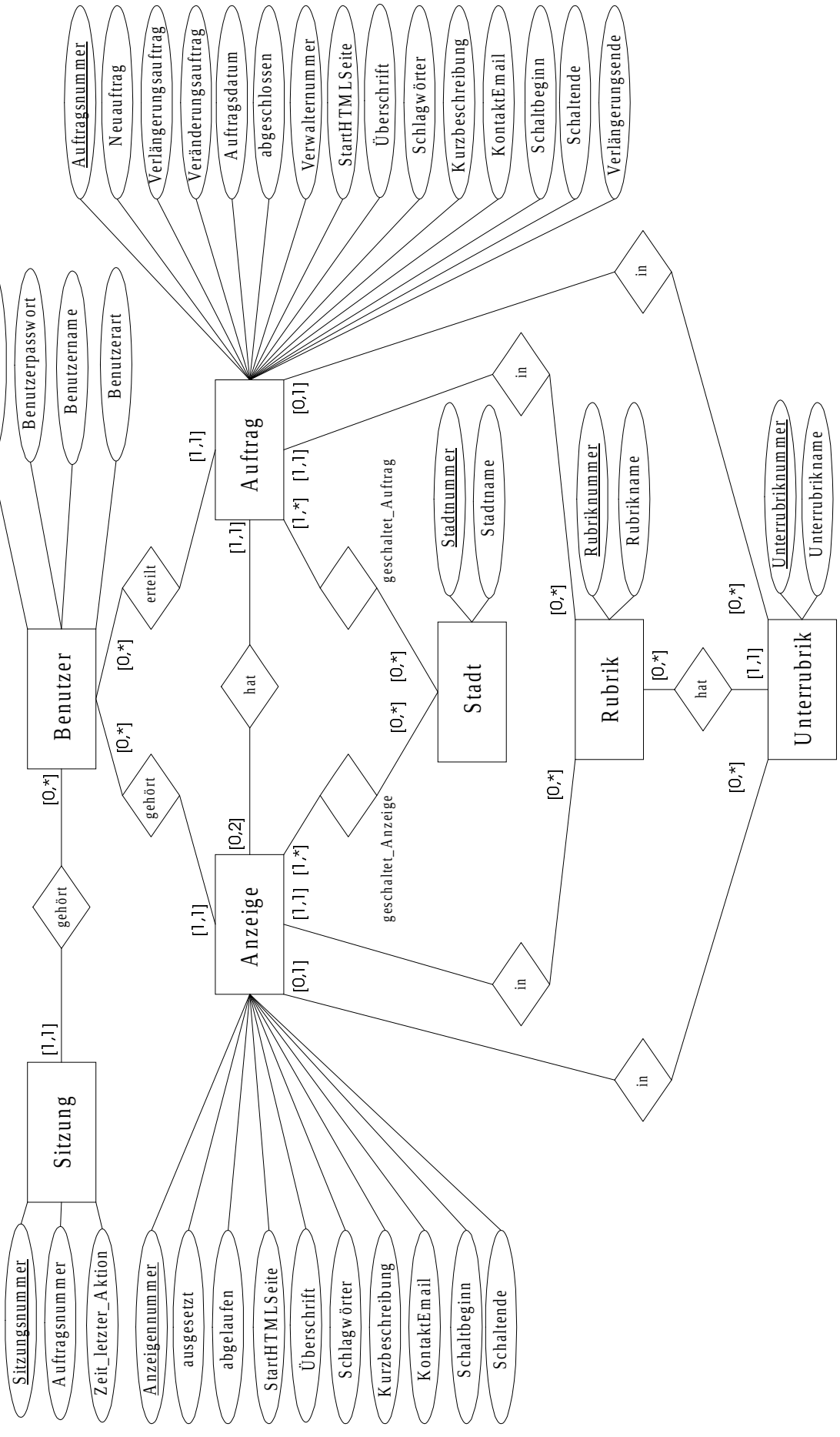

Abb. 31 ER-Modell des ASV-Systems

2. Datenflußdiagramm

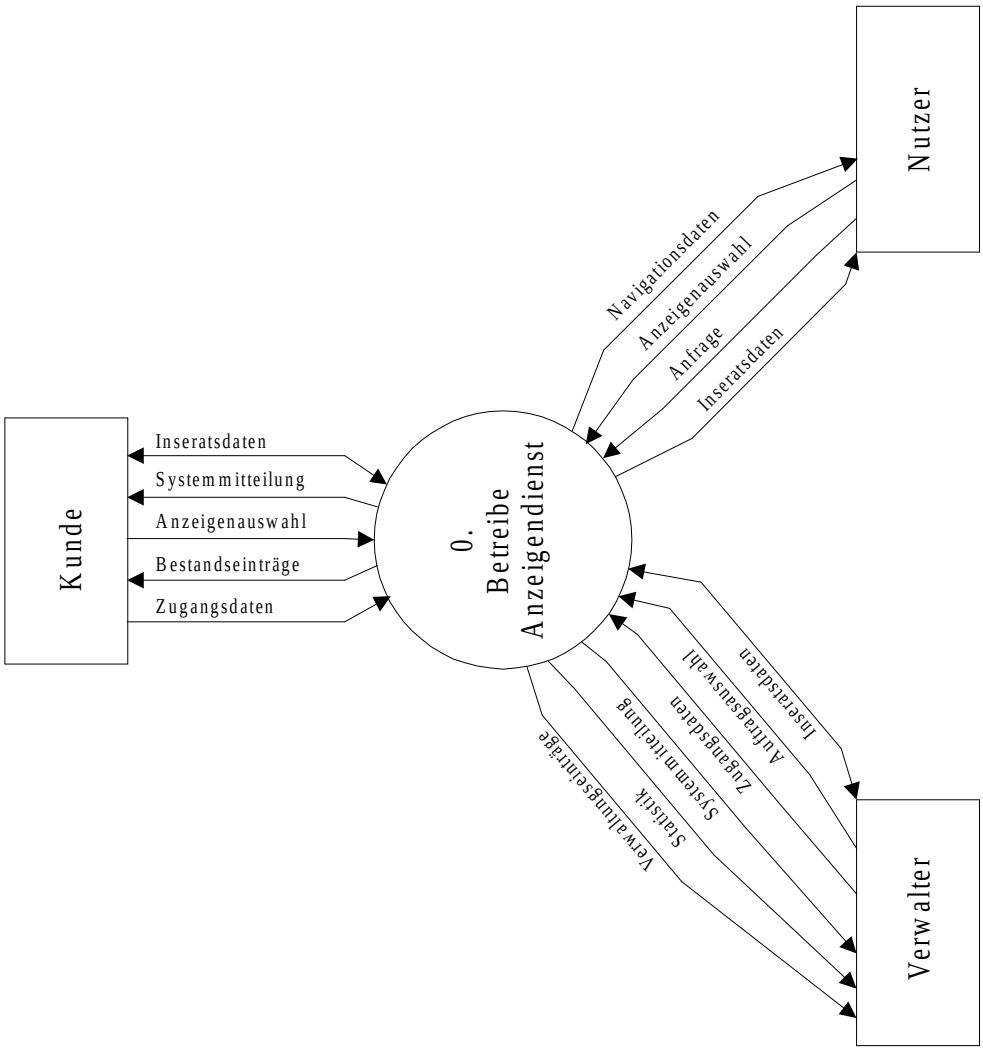

Abb. 32 Kontextdiagramm des ASV-Systems

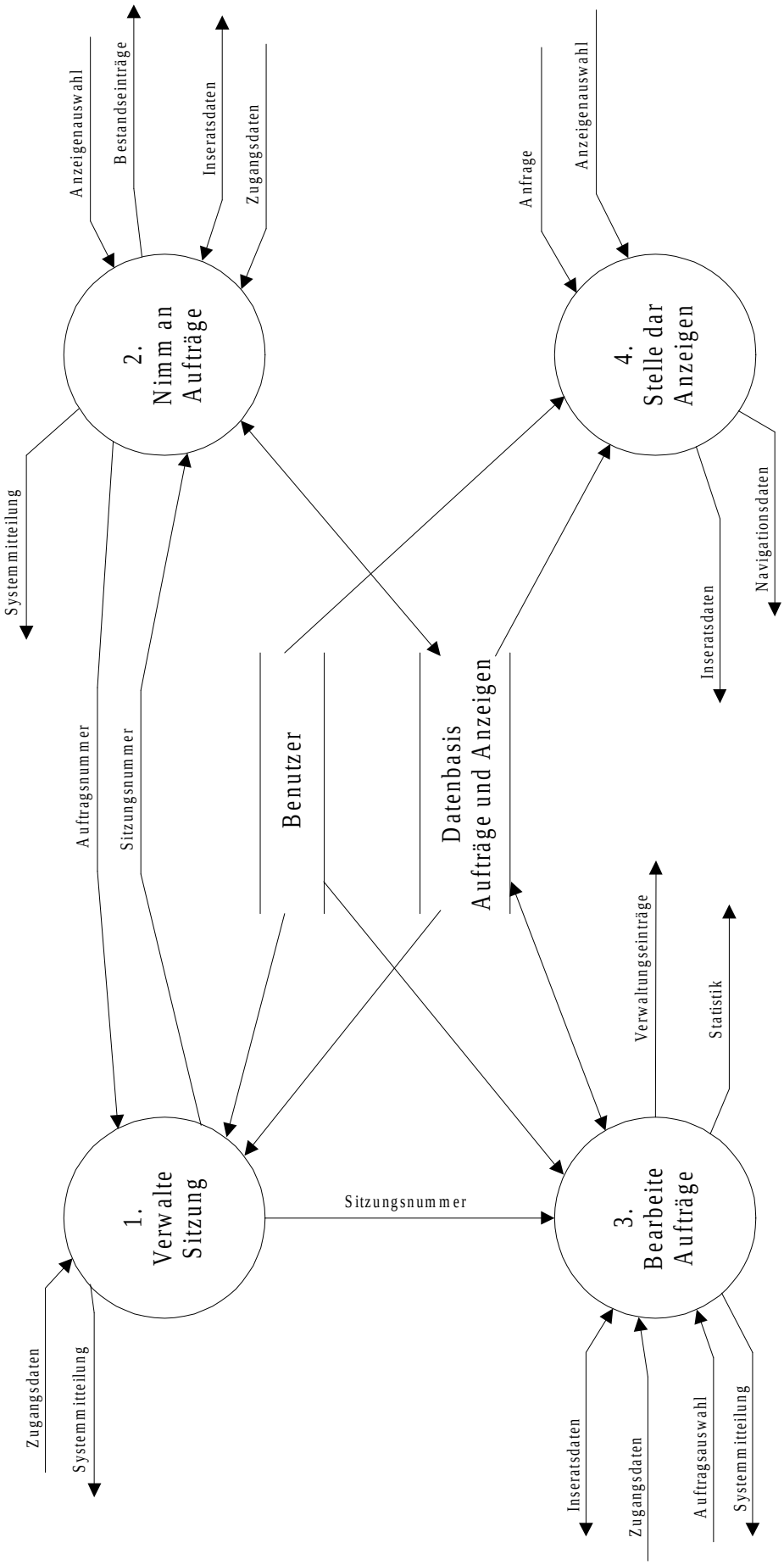

Abb. 33 DFD 0: Betreibe Anzeigendienst

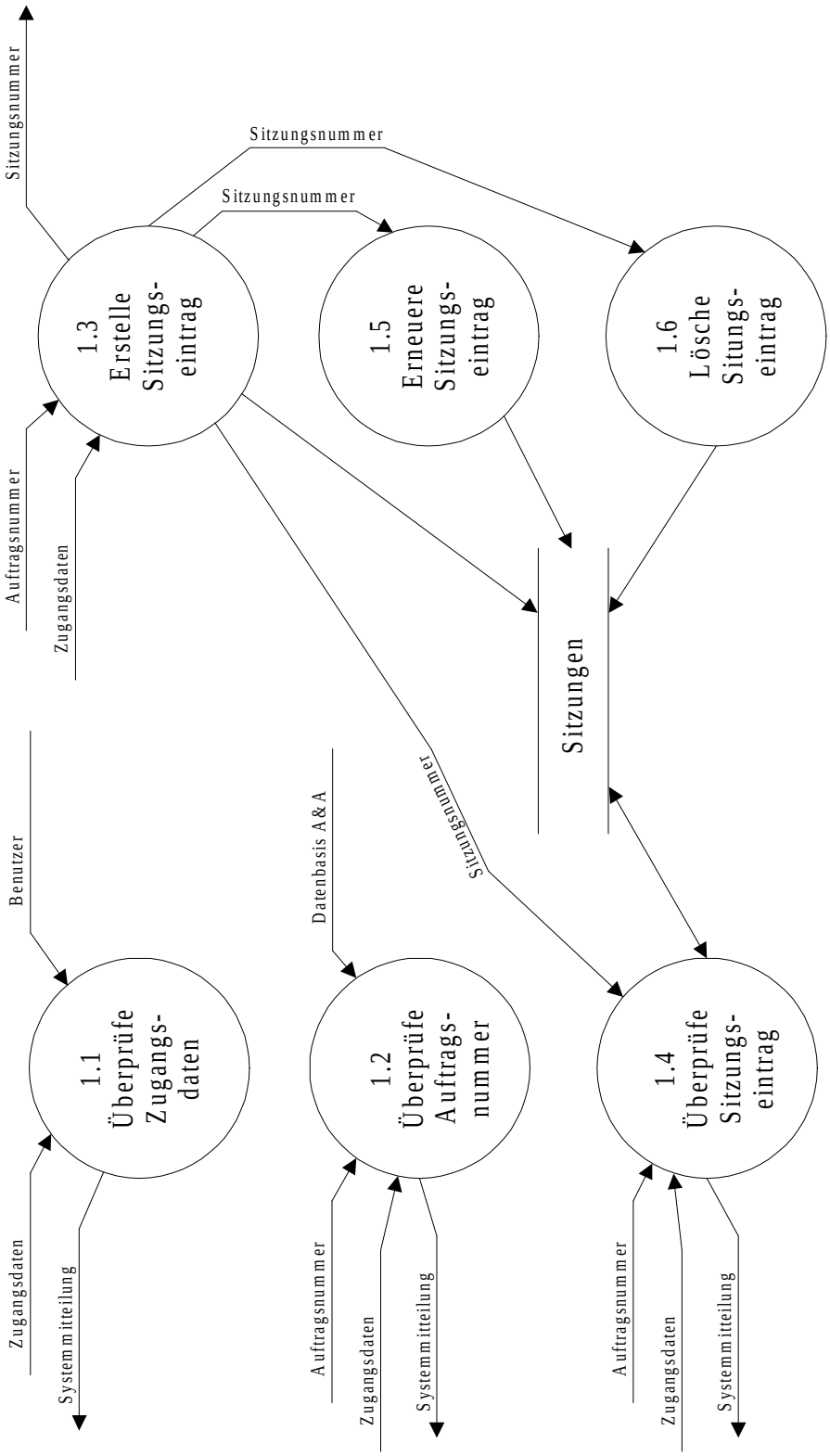

Abb. 34 DFD 1: Verwalte Sitzung

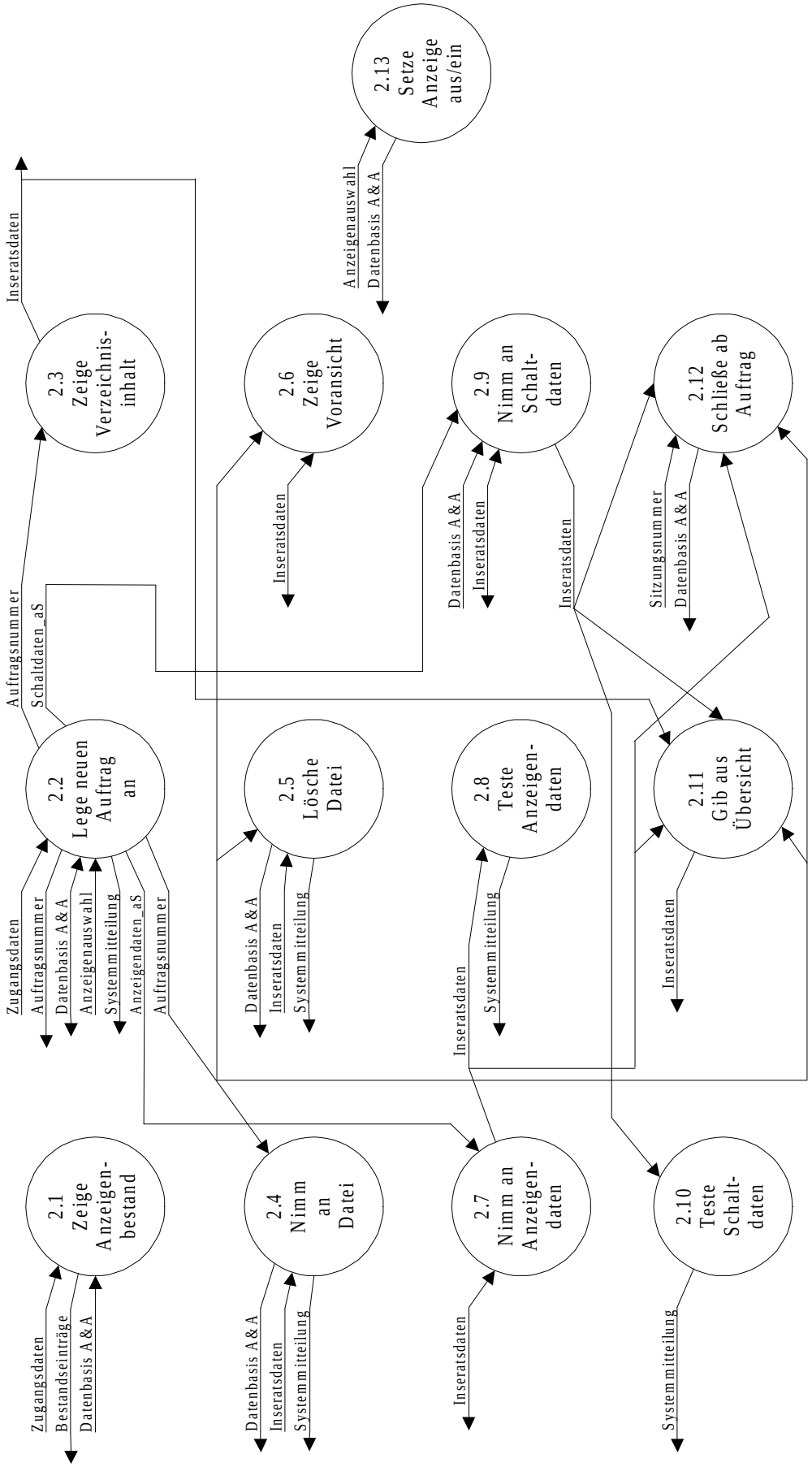

Abb. 35 DFD 2: Nimm an Aufträge

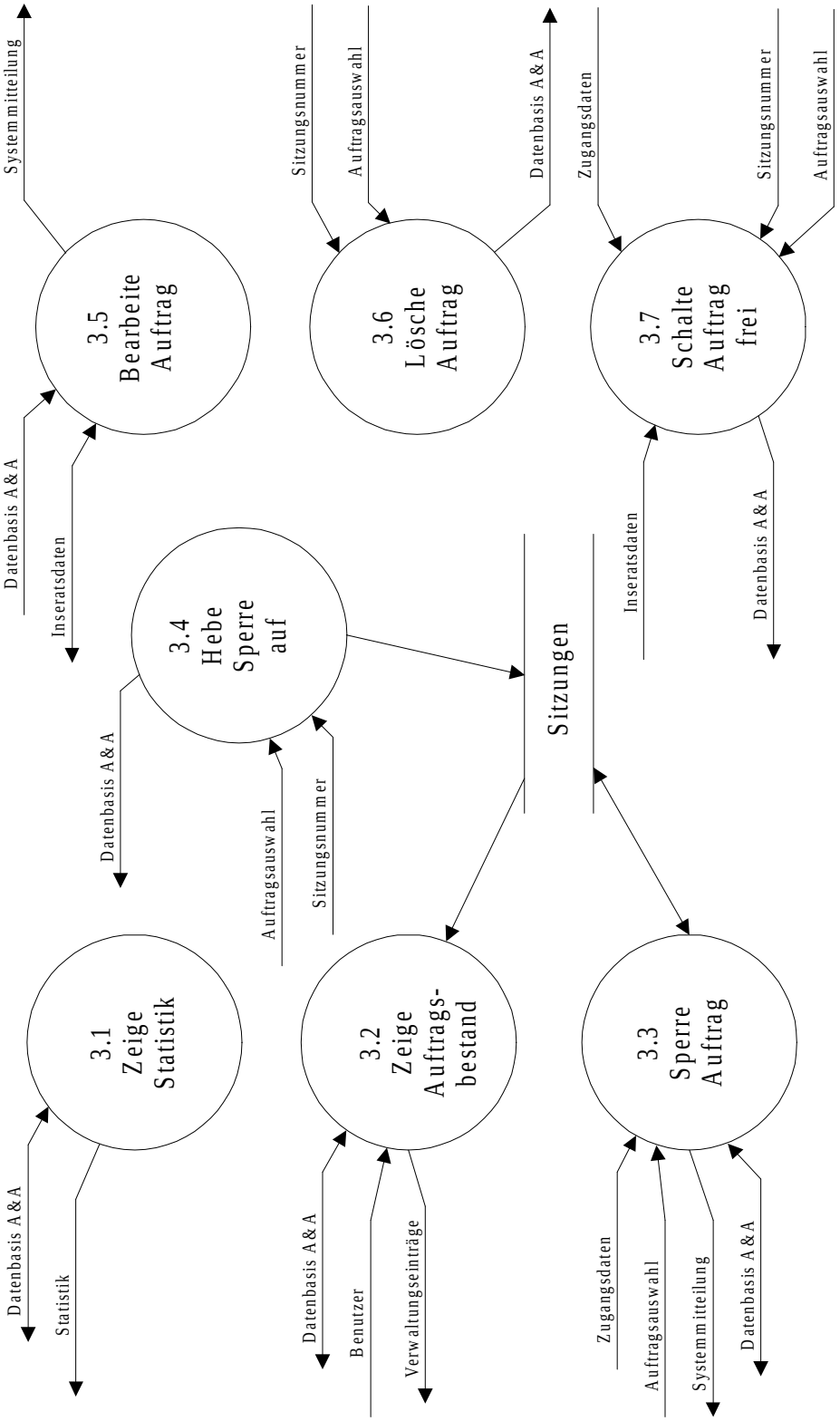

Abb. 36 DFD 3: Bearbeite Aufträge

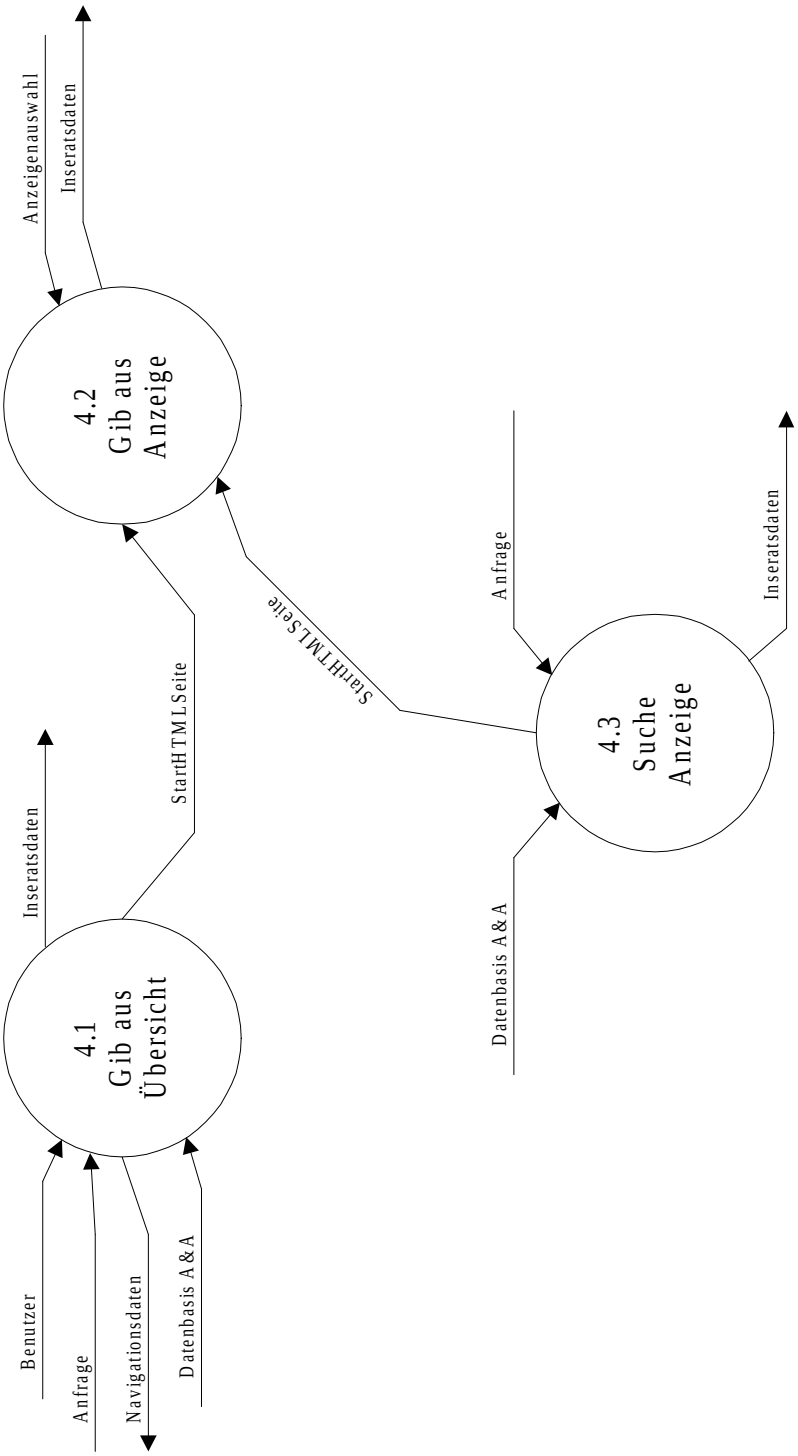

Abb. 37 DFD 4: Stelle dar Anzeigen

3. Pseudocode

Die Numerierung in diesem Abschnitt folgt der Numerierung im DFD.
Mit <*i* ...> werden Paramter der Funktionen und mit <*o* ...> Rückgabewerte der Funktionen bezeichnet.

1. Verwalte Sitzung

Daten allgemein:
<i Auftragsnummer>
<i Datenbasis Aufträge und Anzeigen>
<i Benutzer>
<i Sitzungen>
<i Zugangsdaten>
<o Sitzungen>
    <o Sitzungsnummer>
<o Systemmitteilung>

1.1 Überprüfe Zugangsdaten

Daten:
<i Benutzer>
<i Zugangsdaten>
<o Systemmitteilung>
    <o Zugangsverweigerung>

Pseudocode:
Suche Dateneintrag in <i Benutzer> mit <i Zugangsdaten>
if *not* Dateneintrag gefunden
then
    <o Zugangsverweigerung>;

1.2 Überprüfe Auftragsnummer

Daten:
<i Auftragsnummer>
<i Zugangsdaten>
    <i Benutzerart>
    <i Zugangsnummer>
<i Datenbasis Aufträge und Anzeigen>
<o Systemmitteilung>
    <o Bearbeitungsabbruch>

Pseudocode:
Suche Dateneintrag in <i Datenbasis Aufträge und Anzeigen> mit
<i Auftragsnummer> und <i Zugangsnummer> in (<i Aufträge_Kundennummer>
oder in <i Aufträge_Verwalternummer> in Abhängigkeit von <i Benutzerart>)
if *not* Dateneintrag gefunden
then
    <o Bearbeitungsabbruch>;

1.3 Erstelle Sitzungseintrag

/* vor dem Aufruf dieser Funktion müssen die Zugangsdaten und die Auftragsnummer getestet werden!*/
Daten:
<i Auftragsnummer>
<i Zugangsdaten>
    <i Benutzerart>
    <i Zugangsnummer>
<o Sitzungen>
    <o Sitzungsnummer>

Pseudocode:
Erzeuge <o Sitzungsnummer>;
Erzeuge Datensatz in <o Sitzungen> mit <i Zugangsnummer>,<i Benutzerart>, <i Auftragsnummer>, <o Sitzungsnummer>;

1.4 Überprüfe Sitzungseintrag

Daten:
<i Auftragsnummer>
<i Zugangsdaten>
    <i Benutzerart>
    <i Zugangsnummer>
<i Sitzungen>
    <i Sitzungsnummer>
<o Systemmitteilung>
    <o Bearbeitungsabbruch>
<o Sitzungen>

Pseudocode:
Lösche alle Datensätze in <o Sitzungen> wo <o Sitzungen_Zeit letzter Aktion> länger als festgelegte Anzahl von Minuten her ist;
Suche Dateneintrag in <i Sitzungen> mit <i Zugangsnummer>, <i Benutzerart>, <i Auftragsnummer>, <i Sitzungsnummer>;
if *not* Dateneintrag gefunden
then
    <o Bearbeitungsabbruch>;

1.5 Erneuere Sitzungseintrag

/* Vor dem Aufruf dieser Funktion muß der Sitzungseintrag getestet werden */
Daten:
<i Sitzungsnummer>
<o Sitzungen>

Pseudocode:
Setze <o Sitzungen_Zeit letzter Aktion> auf aktuelle Systemzeit in <o Sitzungen> wo <i Sitzungsnummer>;

1.6 Lösche Sitzungseintrag

/* Vor dem Aufruf dieser Funktion muß der Sitzungseintrag getestet werden */
Daten:
<i Sitzungsnummer>
<o Sitzungen>

Pseudocode:
Lösche Datensatz in <o Sitzungen> wo <i Sitzungsnummer>;

2. nimm an Aufträge

Daten allgemein:
<i Anzeigenauswahl>
<i Datenbasis Aufträge und Anzeigen>
<i Inseratsdaten>
<i Sitzungsnummer>
<i Zugangsdaten>
<o Auftragsnummer>
<o Datenbasis Aufträge und Anzeigen>
<o Inseratsdaten>
<o Systemmitteilung>

2.1 Zeige Anzeigenbestand

Daten:
<i Zugangsdaten>
    <i Zugangsnummer>
<i Datenbasis Aufträge und Anzeigen>
    <i Aufträge>
        <i Auftrag_Auftragsdatum>
        <i Auftrag_Auftragsnummer>
        <i Auftrag_Neuauftrag>
        <i Auftrag_Überschrift>
        <i Auftrag_Verlängerungsauftrag>
        <i Auftrag_Veränderungsauftrag>
    <i Anzeigen>
        <i Anzeige_Anzeigennummer>
        <i Anzeige_Auftragsdatum>
        <i Anzeige_ausgesetzt>
        <i Anzeige_Schaltende>
        <i Anzeige_Überschrift>
<o Bestandseinträge>

Pseudocode:
Suche alle Anzeigen aus <i Anzeigen> für <i Zugangsnummer>;
Suche alle Aufträge aus <i Aufträge> für <i Zugangsnummer>;
Generiere für jede neu geschaltete Anzeige, die noch nicht bearbeitet wurde (<i Auftrag_Neuauftrag>) einen Bestandseintrag mit:

- <i Auftrag_Auftragsnummer>, <i Auftrag_Überschrift>, <i Auftrag_Auftragsdatum>
- dem Infotext: „Dieser Auftrag wird in Kürze bearbeitet. Wir bitten um Ihre Geduld.“;

Generiere für jede bereits einmal freigeschaltete Anzeige einen Bestandseintrag aus:

- <i Anzeige_Anzeigennummer>, <i Anzeige_Überschrift>, <i Anzeige_Auftragsdatum> und der aus <i Anzeige_Schaltende> berechneten Restlaufzeit für eine Anzeige (bzw. den Infotext: „Diese Anzeige ist abgelaufen“)
- den in Abhängigkeit vom Bestehen eines Veränderungs- und/oder Verlängerungsauftrages (<i Auftrag_Veränderungsauftrag>, <i Auftrag_Verlängerungsauftrag>) für diese Anzeige auszugebenden Infotexten für diese Anzeige: „Für diese Anzeige existiert ein [Veränderungs-[und/oder] Verlängerungs]auftrag, der in Kürze bearbeitet wird. Wir bitten um Ihre Geduld.“
- der in Abhängigkeit vom Bestehen eines Veränderungs- und/oder Verlängerungsauftrages für diese Anzeige auszugebenden Funktionsauswahl: „Anzeige [verändern][verlängern]“
- der Funktionsauswahl, die Anzeige aus/einzusetzen: „Anzeige [aus| wieder ein]setzen“ in Abhängigkeit von <i Anzeige_ausgesetzt>;

Stelle <o Bestandseinträge>, gebildet aus allen Bestandseinträgen für Neuaufträge, geordnet nach <i Auftrag_Auftragsdatum> dar;

Stelle <o Bestandseinträge>, gebildet aus allen Bestandseinträgen für schon einmal freigeschaltete Anzeigen, geordnet nach <i Anzeige_Auftragsdatum> dar;

2.2 Lege neuen Auftrag an

Daten:

<i Datenbasis Aufträge und Anzeigen>
    <i Anzeigen>
        <i Anzeige_Anzeigennummer>
        <i Anzeige_KontaktEmail>
        <i Anzeige_Kurzbeschreibung>
        <i Anzeige_Rubriknummer>
        <i Anzeige_Schaltbeginn>
        <i Anzeige_Schaltende>
        <i Anzeige_Schlagwörter>
        <i Anzeige_StartHTMLSeite>
        <i Anzeige_Überschrift>
        <i Anzeige_Unterrubriknummer>
    <i Aufträge>
        <i Auftrag_Anzeigennummer>
        <i Auftrag_Veränderungsauftrag>
        <i Auftrag_Verlängerungsauftrag>
    <i geschaltet_Anzeige>
        <i geschaltet_Anzeige_Anzeigennummer>
        <i geschaltet_Anzeige_Stadtnummer>
<i Anzeigenauswahl>
<i Zugangsdaten>
    <i Benutzerart>

<i Zugangsnummer>
<o Anzeigendaten_aS>
<o Kontakt-Email-Adresse>
<o Kurzbeschreibung>
<o Schlagwörter>
<o Start-HTML-Seite>
<o Überschrift>
<o Datenbasis Aufträge und Anzeigen>
<o Anzeigen>
<o Aufträge>
<o Auftrag_Anzeigennummer>
<o Auftrag_Auftragsnummer>
<o Auftrag_Neuauftrag>
<o Auftrag_Veränderungsauftrag>
<o Auftrag_Verlängerungsauftrag>
<o Systemmitteilung>
<o Auftragsablehnung>
<o Auftragsnummer>
<o Schaltdaten_aS>
<o Schaltbeginn>
<o Schaltende>
<o Rubriknummer>
<o Unterrubriknummer>
<o Stadtnummern>

Pseudocode:

```
/* Überprüfe, ob <i Anzeigenauswahl> zu <i Zugangsnummer> gehört, falls <i
Anzeigenauswahl> vorhanden, so daß keiner für eine fremde Anzeige Aufträge
vergeben kann */
if <i Anzeigenauswahl> // d.h. es ist kein Neuauftrag
then
    Suche Datensatz in <i Anzeigen> wo
    <i Anzeige_Anzeigennummer> == <i Anzeigenauswahl> und
    <i Anzeige_Kundennummer> == <i Zugangsnummer>;
    if not Datensatz gefunden
        then <o Auftragsablehnung>;
/* Überprüfe Vorhandensein von Veränderungs- oder Verlängerungsaufträgen für
die durch <i Anzeigenauswahl> spezifizierte Anzeige bei Veränderungs- oder
Verlängerungsaufträgen */
Suche Datensatz in <i Aufträge> wo <i Auftrag_Anzeigennummer> ==
<i Anzeigenauswahl> und <i Auftrag_Verlängerungsauftrag> == 1 und/oder
<i Auftrag_Veränderungsauftrag> == 1;
if Datensatz vorhanden
then
    lehne Annahme Auftrag ab mit <o Auftragsablehnung>;
// Neuen Auftragsdatensatz in Datenbank anlegen
// Für Neuaufträge:
/* ER-Modell fordert, daß für einen Auftrag bereits eine Anzeige existiert. Bei der
Implementierung wird aber in Tabelle Auftrag stattdessen bei Neuaufträgen der
Null-Wert eingetragen, was eine grosse Vereinfachung bedeutet, da ansonsten ein
```

Anzeigen-Stumpf angelegt werden müßte und dies in weiteren Funktionen ebenfalls beachtet werden müßte.*/
Generiere neue <o Auftragsnummer> und schreibe neuen Datensatz mit
<o Auftrag_Auftragsnummer> = <o Auftragsnummer>,
<o Auftrag_Kundennummer> = <i Zugangsnummer> und belege
<o Auftrag_Neuauftrag> mit 1 sowie <o Auftrag_Verlängerungsauftrag> und
<o Auftrag_Veränderungsauftrag> mit 0;
// Für Veränderungs- oder Verlängerungsaufträge:
Generiere neue <o Auftragsnummer> und schreibe neuen Datensatz mit
<o Auftrag_Auftragsnummer> = <o Auftragsnummer>,
<o Auftrag_Kundennummer> = <i Zugangsnummer>,
<o Auftrag_Anzeigennummer> = <i Anzeigenauswahl> und belege
<o Auftrag_Neuauftrag> mit 0 sowie <o Auftrag_Verlängerungsauftrag> oder <o Auftrag_Veränderungsauftrag> je nach Auftragsart mit 1 bzw. 0;
// Neues Verzeichnis bei Neuauftrag und Veränderungsauftrag
Bei Neuauftrag und Veränderungsauftrag lege neues Verzeichnis mit Bezeichnung <o Auftragsnummer> an und mache Verzeichnis lesbar für alle;
Bei Veränderungsauftrag kopiere Dateien aus Verzeichnis <i Auftrag_Anzeigennummer> in das neuangelegte Verzeichnis;
// Neue Sitzung generieren
Veranlasse neue Sitzung durch Aufruf von F „erstelle Sitzungseintrag“ mit Übergabe von <o Auftragsnummer>, <i Zugangsnummer> und <i Benutzerart>;
// Lade Anzeigendaten bei Veränderungs- oder Verlängerungsaufträgen
Lade <o Start-HTML-Seite>, <o Überschrift>, <o Schlagwörter>,
<o Kurzbeschreibung>, <o Kontakt-Email-Adresse>, <o Schaltbeginn>,
<o Schaltende>, <o Rubriknummer>, <o Unterrubriknummer>
mit
<i Anzeige_StartHTMLSeite>, <i Anzeige_Überschrift>,
<i Anzeige_Schlagwörter>, <i Anzeige_Kurzbeschreibung>,
<i Anzeige_KontaktEmail>, <i Anzeige_Schaltbeginn>, <i Anzeige_Schaltende>,
<i Anzeige_Rubriknummer>, <i Anzeige_Unterrubriknummer> aus <i Anzeigen>
wo <i Anzeige_Anzeigennummer> == <i Anzeigennummer>;
Lade <o Stadtnummern> mit <i geschaltet_Anzeige_Stadtnummer> wo
<i geschaltet_Anzeige_Anzeigennummer> == <i Anzeigennummer>;

2.3 Zeige Verzeichnisinhalt

Daten:
<i Auftragsnummer>
< Inseratsdaten>
    <o Verzeichnisinhalt>

Pseudocode:
Lies alle Dateinamen und die dazugehörige Dateigröße aus dem Verzeichnis
<i Auftragsnummer>;
Gib aus <o Verzeichnisinhalt>;

2.4 Nimm an Datei

Daten:
<i Inseratsdaten>
    <i Anzeigendateien>
        <i Anzeigendatei>
<i Auftragsnummer>
<o Datenbasis Aufträge und Anzeigen>
    <o Anzeigendateien>
<o Systemmitteilung>
    <o Dateiablehnung>

Pseudocode:
Biete Dateiauswahlbox;
if Datei ausgewählt und per Browser an Server abgeschickt
then
    Verschiebe hochgeladene Datei <i Anzeigendatei> aus dem temporären Hochladeverzeichnis in das Verzeichnis mit Namen <i Auftragsnummer>, schreibe also in <o Anzeigendateien>;
    Vergib Leserechte für alle an Datei <i Anzeigendatei>;
if Größe aller Dateien in Verzeichnis <i Auftragsnummer> größer als vorgegebener Wert
then
    Lösche <i Anzeigendatei>;
    <o Dateiablehnung>;

2.5 Lösche Datei

Daten:
<i Inseratsdaten>
    <i Anzeigendateien>
        <i Anzeigendatei>
<i Auftragsnummer>
<o Datenbasis Aufträge und Anzeigen>
    <o Anzeigendateien>
<o Systemmitteilung>
    <o Löschablehnung>

Pseudocode:
Biete Dateiauswahlbox;
if Datei ausgewählt
then
    Prüfe, ob Datei <i Anzeigendatei> in Verzeichnis <i Auftragsnummer> existiert.
    if existiert
    then
        Lösche Datei <i Anzeigendatei>, schreibe also
        <o Anzeigendateien>;
    else
        <o Löschablehnung>;

2.6 Zeige Voransicht

Daten:
<i Auftragsnummer>
<i Inseratsdaten>
    <i Anzeigendateien>
    <i Anzeigendaten>
        <i Start-HTML-Seite>

<o Inseratsdaten>
    <o Anzeigendateien>

Pseudocode:
Lade <i Start-HTML-Seite> aus Verzeichnis <i Auftragsnummer>
(<i Anzeigendateien>) und zeige sie an (<o Anzeigendateien>);

2.7 Nimm an Anzeigendaten

Daten:
<i Inseratsdaten>
    <i Anzeigendaten>
<i Anzeigendaten_aS>
<o Inseratsdaten>
    <o Anzeigendaten>

Pseudocode:
if <i Anzeigendaten_aS>
then
    Fülle Eingabemaske mit <i Anzeigendaten_aS>;
Nimm <i Anzeigendaten> über Eingabemaske entgegen und gib sie als
<o Anzeigendaten> weiter;

2.8 Teste Anzeigendaten

Daten:
<i Inseratsdaten>
    <i Anzeigendaten>
<o Systemmitteilung>
    <o Eingabeberichtigung>

Pseudocode:
Teste <i Anzeigendaten> nach Vorgaben;
if Fehler
    then <o Eingabeberichtigung>;

2.9 Nimm an Schaltdaten

Daten:

<i Datenbasis Aufträge und Anzeigen>
    <i Rubriken>
    <i Schaltdaten>
    <i Städte>
    <i Unterrubriken>
<i Schaltdaten_aS>
<o Inseratsdaten>
    <o Schaltdaten>

Pseudocode:
Lade alle Rubriknummern und -namen aus <i Rubriken>;
Lade alle Stadtnummern und -namen aus <i Städte>;
Lade alle Unterrubriknummern und -namen aus <i Unterrubriken>;
if <i Schaltdaten_aS>
then
    Fülle Eingabemaske mit <i Schaltdaten_aS> und geladenen Rubriken-, Unterrubrik-und Stadtnamen;
    Markiere die Rubrik, die Unterrubrik und die Stadtnamen, die in <i Schaltdaten_aS> vorgegeben sind;
Nimm <i Schaltdaten> über Eingabemaske entgegen und gib sie als <o Schaltdaten> weiter;

2.10 Teste Schaltdaten

Daten:
<i Inseratsdaten>
    <i Schaltdaten>
<o Systemmitteilung>
    <o Eingabeberichtigung>

Pseudocode:
Teste <i Schaltdaten> nach Vorgaben;
if Fehler
    then <o Eingabeberichtigung>;

2.11 Gib aus Übersicht

Daten:
<i Inseratsdaten>
    <i Anzeigendaten>
    <i Schaltdaten>
    <i Verzeichnisinhalt>
<i Auftragsnummer>
<o Inseratsdaten>

Pseudocode:
Generiere aus <i Schaltdaten>, <i Anzeigendaten> und den Namen der Dateien aus dem Verzeichnis mit Namen <i Auftragsnummer> (<i Verzeichnisinhalt>) eine Übersicht (<o Inseratsdaten>);

2.12 Schließe ab Auftrag

Daten:
<i Inseratsdaten>
    <i Anzeigendaten>
    <i Schaltdaten>
        <i Stadtnummern>
<i Auftragsnummer>
<i Sitzungsnummer>
<o Datenbasis Aufträge und Anzeigen>
    <o Aufträge>
    <o geschaltet_Auftrag>

Pseudocode:
Je nach Auftragsart speichere <i Anzeigendaten> und <i Schaltdaten > in
<o Aufträge> mit <i Auftragsnummer>, setze <o Auftrag_abgeschlossen> in jedem Fall auf 1;
Speichere <i Stadtnummern> in <o geschaltet_Auftrag> für <i Auftragsnummer> bei Neu- und Veränderungsaufträgen;
Sperre Verzeichnis mit Namen <i Auftragsnummer> für Zugriff von außen bei Neu- und Veränderungsaufträgen;
Beende Sitzung durch Aufruf von F „lösche Sitzungseintrag“ mit
<i Sitzungsnummer>;
Gib Bestätigungsseite aus;

2.13 Setze Anzeige aus/ein

/* Vor dem Aufrufen dieser Funktion muß eine Sitzung eröffnet werden, wobei die Zugehörigkeit von <i Anzeigenauswahl> und Zugangsnummer geprüft wird. */
Daten:
<i Anzeigenauswahl>
<o Datenbasis Aufträge und Anzeigen>
    <o Anzeigen>
        <o Anzeige_ausgesetzt>

Pseudocode:
Setze <o Anzeige_ausgesetzt> auf [1|0] wo Anzeigennummer =
<i Anzeigenauswahl>;

3. Bearbeite Aufträge

Daten allgemein:
<i Auftragsauswahl>
<i Datenbasis Aufträge und Anzeigen>
<i Benutzer>
<i Inseratsdaten>
<i Sitzungen>
    <i Sitzungsnummer>
<i Zugangsdaten>
<o Datenbasis Aufträge und Anzeigen>
<o Inseratsdaten>
<o Sitzungen>
<o Statistik>
<o Verwaltungseinträge>
<o Systemmitteilung>

3.1 Zeige Statistik

Daten:
<i Datenbasis Aufträge und Anzeigen>
    <i Aufträge>
<o Datenbasis Aufträge und Anzeigen>
    <o Anzeigen>
<o Statistik>

Pseudocode:
Markiere alle Anzeigen <o Anzeigen> als abgelaufen, wo Schaltende kleiner als aktuelle Systemzeit, merke Anzahl;
Ermittle aus <i Aufträge> die Anzahl der Neu-, Veränderungs- und Verlängerungsaufträge und gib sie zusammen mit der Anzahl der abgelaufenen Anzeigen als <o Statistik> aus;

3.2 Zeige Auftragsbestand

Daten:
<i Datenbasis Aufträge und Anzeigen>
    <i Aufträge>
        <i Auftrag_Verwalternummer>
<i Benutzer>
    <i Benutzername>
    <i Benutzerart>
<i Sitzungen>
<o Datenbasis Aufträge und Anzeigen>
    <o Aufträge>
        <o Auftrag_Verwalternummer>
<o Verwaltungseinträge>

Pseudocode:
Lösche alle <o Auftrag_Verwalternummer> Einträge in Aufträgen, wenn für diesen Auftrag und diese Verwalternummer kein Eintrag in <i Sitzungen>;
Je nach gewünschter Auftragsart (Neu-, Veränderungs-, Verlängerungsauftrag) suche entsprechende Aufträge heraus und stelle Verwaltungseintrag aus
<i Aufträge>, <i Benutzer> zusammen mit
- Auftragsüberschrift und Kundenname

if in <i Auftrag_Verwalternummer> kein Eintrag vorhanden
then mit
- Funktionsauswahl „sofort freischalten“, „bearbeiten“, „verwerfen“;

else mit // Auftrag ist also in Bearbeitung durch einen Verwalter
- „Auftrag ist gerade in Bearbeitung durch Verwalter Verwaltername“;

Fasse Verwaltungseinträge zu <o Verwaltungseinträge> zusammen und gib aus;

3.3 Sperre Auftrag

Daten:
<i Auftragsauswahl>
<i Datenbasis Aufträge und Anzeigen>
    <i Aufträge>
        <i Auftrag_Verwalternummer>
<i Sitzungen>
<i Zugangsdaten>
    <i Benutzerart>
    <i Zugangsnummer>
<o Datenbasis Aufträge und Anzeigen>
    <o Aufträge>
        <o Auftrag_Verwalternummer>
<o Sitzungen>
<o Systemmitteilung>
    <o Bearbeitungsabbruch>

Pseudocode:
// Sperrung setzen
Setze <o Auftrag_Verwalternummer> auf <i Zugangsnummer> in Auftrag <i Auftragsauswahl> falls <i Auftrag_Verwalternummer> == 0;
if Sperrung erfolgreich
then
    // Neue Sitzung generieren
    Veranlasse neue Sitzung durch Aufruf von F „erstelle Sitzungseintrag“ mit Übergabe von <o Auftragsauswahl>, <i Zugangsnummer> und <i Benutzerart>;
else // in <i Auftrag_Verwalternummer> muß schon eine Sperrung existieren
    // Überprüfe, ob Verwalter, der Sperrung innehat, ein anderer ist
    if <i Auftrag_Verwalternummer> == <i Zugangsnummer>
    then // Sperrung durch den gleichen Verwalter
        Stelle Verwalter vor Möglichkeit, auf der Bearbeitung zu bestehen oder aber auch den Auftrag freizugeben;
        if Verwalter besteht auf Bearbeitung
        then
            // Alte Sitzung löschen

Finde Sitzungsnummer der eigenen anderen Sitzung in
<i Sitzungen> heraus;
Lösche diese Sitzung durch Auftruf von F „lösche Sitzungseintrag“;
// Neue Sitzung generieren
Veranlasse neue Sitzung durch Aufruf von F „erstelle Sitzungseintrag“ mit Übergabe von <o Auftragsauswahl>, <i Zugangsnummer> und <i Benutzerart>;

elseif Verwalter gibt Sitzung frei
Setze <o Auftrag_Verwalternummer> auf 0 für
<i Auftragsauswahl>;
// Alte Sitzung löschen
Finde Sitzungsnummer der eigenen anderen Sitzung in
<i Sitzungen> heraus;
Lösche diese Sitzung durch Auftruf von F „lösche Sitzungseintrag“;

else // es ist ein anderer
// Test, ob anderer Verwalter noch eingeloggt ist
if Eintrag in <i Sitzungen> mit <i Auftrag_Verwalternummer> und
<i Auftragsauswahl>
then
<o Bearbeitungsabbruch>;
F zeige Auftragsbestand; // aktualisiere Anzeige
else
// Sperrung an sich reissen
Setze <o Auftrag_Verwalternummer> auf
<i Zugangsnummer> in Auftrag <i Auftragsauswahl>;
// Neue Sitzung generieren
Veranlasse neue Sitzung durch Aufruf von F „erstelle Sitzungseintrag“ mit Übergabe von <o Auftragsauswahl>, <i Zugangsnummer> und <i Benutzerart>;

3.4 Hebe Sperre auf

Daten:
<i Auftragsauswahl>
<i Sitzungsnummer>
<o Datenbasis Aufträge und Anzeigen>
<o Aufträge>
<o Auftrag_Verwalternummer>
<o Sitzungen>

Pseudocode:
// Sperrung aufheben
Setze <o Auftrag_Verwalternummer> auf 0 in Auftrag <i Auftragsauswahl>;
// Sitzung beenden
Veranlasse Beendigung der Sitzung in <o Sitzungen> durch Aufruf von F „lösche Sitzungseintrag“ mit Übergabe vin <i Sitzungsnummer>;

3.5 Bearbeite Auftrag

Daten:
<i Datenbasis Aufträge und Anzeigen>
    <i Aufträge>
<i Inseratsdaten>
<o Inseratsdaten>
<o Systemmitteilung>

Pseudocode:
// Funktion wird nicht weiter verfeinert, da sie bereits folgende unter „Nimm an Aufträge“ definierten Funktionen verwendet:
- „Nimm an Dateien“
- „Zeige Verzeichnisinhalt“
- „Lösche Datei“
- „Nimm an Anzeigendaten“
- „Teste Anzeigendaten“
- „Nimm an Schaltdaten“
- „Teste Schaltdaten“
- „Zeige Voransicht“

3.6 Lösche Auftrag

Daten:
<i Auftragsauswahl>
<i Sitzungsnummer>
<o Datenbasis Aufträge und Anzeigen>
    <o Aufträge>
    <o geschaltet_Auftrag>

Pseudocode:
// Auftrag löschen
Lösche in <o Aufträge>, wo Auftragsnummer == <i Auftragsauswahl>;
// lösche Eintrag in Tabelle ‚geschaltet‘ für diesen Auftrag
Lösche in <o geschaltet_Auftrag>, wo Auftragsnummer == <i Auftragsauswahl>;
// Bei Neu- oder Veränderungsaufträgen lösche Verzeichnis
Lösche Verzeichnis Aufträge/<i Auftragsauswahl>;
// Sitzung beenden
Veranlasse Beendigung der Sitzung in <o Sitzungen> durch Aufruf von F „lösche Sitzungseintrag“ mit Übergabe von <i Sitzungsnummer>;

3.7 Schalte Auftrag frei

Daten:
<i Auftragsauswahl>
<i Inseratsdaten>
    <i Anzeigendaten>
    <i Schaltdaten>
        <i Stadtnummern>
<i Sitzungsnummer>

<i Zugangsdaten>
    <i Zugangsnummer>
<o Datenbasis Aufträge und Anzeigen>
    <o Anzeigen>
    <o geschaltet_Anzeige>
    <o Aufträge>

Pseudocode:
// Speichere Inseratsdaten für Anzeige
// Für Veränderungs- und Verlängerungsaufträge:
Ermittle Anzeigennummer <Anz-Nr.>, die <i Auftragsauswahl> betrifft;
// Speichere Anzeigen- und Schaltdaten

// Für Neuauftrag:
Speichere <i Anzeigendaten> und <i Schaltdaten> in <o Anzeigen> mit Anzeigennummer = <i Auftragsauswahl>;
Speichere <i Stadtnummern> in <o geschaltet_Anzeige> mit Anzeigennummer = <i Auftragsauswahl>;
// Sonst:
Speichere <i Anzeigendaten> und <i Schaltdaten> in <o Anzeigen>, wo Anzeigennummer == ermittelte Anz-Nr.;
Speichere <i Stadtnummern> in <o geschaltet_Anzeige> wo Anzeigennummer == ermittelte Anz-Nr. ;
// Sichere Anzeigendateien bei Neu- oder Veränderungsauftrag
// Für einen Neuauftrag:
Erstelle ein Verzeichnis Anzeigen/<i Auftragsauswahl>;
// Für Neu- oder Veränderungsauftrag:
Kopiere Verzeichnisinhalt von Verzeichnis Aufträge/<i Auftragsauswahl>/ nach Anzeigen/<Anz-Nr.>; Lösche nun Verzeichnis Aufträge/<i Auftragsauswahl>;
// lösche Auftrag
Veranlasse Löschung des Auftrages durch Aufruf von F „lösche Auftrag“ mit Übergabe von <i Auftragsauswahl>, <i Sitzungsnummer>;

4. Stelle dar Anzeigen

Daten allgemein:
<i Anfrage>
<i Anzeigenauswahl>
<i Datenbasis Aufträge und Anzeigen>
<i Benutzer>
<o Inseratsdaten>
<o Navigationsdaten>

4.1 Gib aus Übersicht

Daten:
<i Anfrage>
    <i Kundennummer>
    <i Rubriknummer>

<i Stadtnummer>
<i Unterrubriknummer>
<i Datenbasis Aufträge und Anzeigen>
<i Anzeigen>
<i geschaltet_Anzeige>
<i Rubriken>
<i Städte>
<i Unterrubriken>
<i Benutzer>
<i Zugangsnummer>
<o Inseratsdaten>
<o Navigationsdaten>
<o Kundeneinträge>
<o Rubrikeinträge>
<o Stadteinträge>
<o Unterrubrikeinträge>
<o StartHTMLSeite>

Pseudocode:
if Ausgabe nach Rubriken
then
Biete Auswahlmöglichkeit der Rubrik/Unterrubrik/Städte durch Lesen von <i Rubriken>,<i Unterrubriken>,<i geschaltet_Anzeige> und <i Städte> und Ausgabe von <o Rubrikeinträge>,<o Unterrubrikeinträge>, <o Stadteinträge>;
Suche alle Anzeigen aus <i Anzeigen> mit <i Rubriknummer> oder mit <i Rubriknummer> und <i Unterrubriknummer> oder mit <i Rubriknummer>, <i Unterrubriknummer> und <i Stadtnummer> und gib ihre <o Inseratsdaten> aus;

else // also Ausgabe nach Kundennamen
Biete Auswahlmöglichkeit unter <o Kundeneinträge>;
Suche alle Anzeigen aus <i Anzeigen> mit <i Zugangsnummer>;

Gib Anzeigen geordnet nach Auftragsdatum aus, stelle <o Start-HTML-Seite> für jede Anzeige bereit;

4.2 Gib aus Anzeige

Daten:
<i Anzeigenauswahl>
<i Start-HTML-Seite>
<o Inseratsdaten>
<o Anzeigendateien>

Pseudocode:

Gib aus Datei <i Start-HTML-Seite> aus Verzeichnis <i Anzeigennummer> als

<o Anzeigendateien>;

4.3 Suche Anzeige

Daten:
<i Anfrage>
    <i Schlagwörter>
<i Datenbasis Aufträge und Anzeigen>
    <i Anzeigen>
    <i Rubriken>
<o Inseratsdaten>
<o Start-HTML-Seite>

Pseudocode:
Suche Anzeigen in <i Anzeigen> mit <i Schlagwörter>;
Gib aus Anzeigen (<o Inseratsdaten>) geordnet nach Rubriken und Auftragsdatum, stelle bereit <o Start-HTML-Seite>;

4. Data Dictionary

Die Syntax und Bezeichnungen des Data Dictionary folgen der im CASE-Tool *INNOVATOR* des Unternehmens *MID GmbH* benutzten Syntax und Bezeichnungen. *D* steht für *Data* und *V* steht für *Views*. Mit *V* bezeichnete Daten werden in der Datenbank gespeichert.

D Anfrage = [Kundennummer | (Rubriknummer + (Unterrubriknummer) + (Stadtnummer)) | ‚Schlagwörter‘].

D Anzeigenauswahl = Anzeigennummer.

D Anzeigendaten_aS = Anzeigendaten.

D Anzeigennummer = ZAHL.
D Auftragsauswahl = Auftragsnummer.
D Auftragsnummer = ZAHL.

V Benutzer = Benutzer_Zugangsnummer +
Benutzer_Benutzerpasswort +
Benutzer_Benutzername +
Benutzer_Benutzerart.
V Benutzer_Zugangsnummer = ZAHL.
V Benutzer_Benutzerpasswort = STRING.
V Benutzer_Benutzername = STRING.
V Benutzer_Benutzerart = STRING.

D ‚Datenbasis Aufträge und Anzeigen‘ = Anzeigendateien + ‚Aufträge‘ +
Anzeigen + Rubriken + Unterrubriken + geschaltet_Anzeige +
geschaltet_Auftrag + ‚Städte‘.
D Anzeigendateien = 1{Anzeigendatei}.
D Anzeigendatei = DATEI.
V ‚Aufträge‘ = Auftrag_Auftragsnummer +
Auftrag_Neuauftrag +
'Auftrag_Verlängerungsauftrag' +
'Auftrag_Veränderungsauftrag' +
Auftrag_Kundennummer +
Auftrag_abgeschlossen +
Auftrag_Auftragsdatum +
Auftrag_Anzeigennummer +
Auftrag_StartHTMLSeite +
'Auftrag_Überschrift' +
'Auftrag_Schlagwörter' +
Auftrag_Kurzbeschreibung +
Auftrag_KontaktEmail +
Auftrag_Rubriknummer +
Auftrag_Unterrubriknummer +
Auftrag_Verwalternummer +
Auftrag_Schaltbeginn +
Auftrag_Schaltende +

'Auftrag_Verlängerungsende'.
V Auftrag_Auftragsnummer = ZAHL.
V Auftrag_Neuauftrag = BOOLEAN.
V 'Auftrag_Verlängerungsauftrag' = BOOLEAN.
V 'Auftrag_Veränderungsauftrag' = BOOLEAN.
V Auftrag_Kundennummer = ZAHL.
V Auftrag_abgeschlossen = BOOLEAN.
V Auftrag_Auftragsdatum = DATE.
V Auftrag_Anzeigennummer = ZAHL.
V Auftrag_StartHTMLSeite = STRING.
V 'Auftrag_Überschrift' = STRING.
V 'Auftrag_Schlagwörter' = STRING.
V Auftrag_Kurzbeschreibung = TEXT.
V Auftrag_KontaktEmail = STRING.
V Auftrag_Rubriknummer = ZAHL.
V Auftrag_Unterrubriknummer = ZAHL.
V Auftrag_Verwalternummer = ZAHL.
V Auftrag_Schaltbeginn = DATE.
V Auftrag_Schaltende = DATE.
V 'Auftrag_Verlängerungsende' = DATE.
V Anzeigen = Anzeige_Unterrubriknummer +
Anzeige_Anzeigennummer +
Anzeige_StartHTMLSeite +
'Anzeige_Überschrift' +
'Anzeige_Schlagwörter' +
Anzeige_Kurzbeschreibung +
Anzeige_KontaktEmail +
Anzeige_Schaltbeginn +
Anzeige_Schaltende +
Anzeige_Kundennummer +
Anzeige_ausgesetzt +
Anzeige_abgelaufen +
Anzeige_Rubriknummer.
V Anzeige_Unterrubriknummer = ZAHL.
V Anzeige_Anzeigennummer = ZAHL.
V Anzeige_StartHTMLSeite = STRING.
V 'Anzeige_Überschrift' = STRING.
V 'Anzeige_Schlagwörter' = STRING.
V Anzeige_Kurzbeschreibung = TEXT.
V Anzeige_KontaktEmail = STRING.
V Anzeige_Schaltbeginn = DATE.
V Anzeige_Schaltende = DATE.
V Anzeige_Kundennummer = ZAHL.
V Anzeige_ausgesetzt = BOOLEAN.
V Anzeige_abgelaufen = BOOLEAN.
V Anzeige_Rubriknummer = ZAHL.
V geschaltet_Anzeige = geschaltet_Stadtnummer +
geschaltet_Anzeigennummer.
V geschaltet_Stadtnummer = ZAHL.
V geschaltet_Anzeigennummer = ZAHL.

V geschaltet_Auftrag = geschaltet_Stadtnummer +
geschaltet_Auftragsnummer.
V geschaltet_Stadtnummer = ZAHL.
V geschaltet_Auftragsnummer = ZAHL.
V Rubriken = Rubrik_Rubriknummer +
Rubrik_Rubrikname.
V Rubrik_Rubriknummer = ZAHL.
V Rubrik_Rubrikname = STRING.
V Unterrubriken = Unterrubrik_Rubriknummer +
Unterrubrik_Unterrubriknummer +
Unterrubrik_Unterrubrikname.
V Unterrubrik_Rubriknummer = ZAHL.
V Unterrubrik_Unterrubriknummer = ZAHL.
V Unterrubrik_Unterrubrikname = STRING.
V 'Städte' = Stadt_Stadtnummer +
Stadt_Stadtname.
V Stadt_Stadtnummer = ZAHL.
V Stadt_Stadtname = STRING.

D Inseratsdaten = Anzeigendateien + Verzeichnisinhalt + Anzeigendaten +
Schaltdaten.
D Anzeigendateien = 1{Anzeigendatei}.
D Anzeigendatei = DATEI.
D Verzeichnisinhalt = {Dateiname+‘Dateigröße‘}.
D Dateiname = STRING.
D ‚Dateigröße‘ = ZAHL.
D Anzeigendaten = ‚Start-HTML-Seite‘ + ‘Überschrift‘ +
(‚Schlagwörter‘) + (Kurzbeschreibung) +
‚Kontakt-Email-Adresse‘.
D ‚Start-HTML-Seite‘ = STRING.
D ‚Überschrift‘ = STRING.
D ‚Schlagwörter‘ = TEXT.
D ‚Kurzbeschreibung‘ = TEXT.
D ‚Kontakt-Email-Adresse‘ = STRING.
D Schaltdaten = Rubriknummer + (Unterrubriknummer) + Stadtnummern+
Schaltbeginn + Schaltende.
D Rubriknummer = ZAHL.
D Unterrubriknummer = ZAHL.
D Stadtnummern = ARRAY OF ZAHL.
D Schaltbeginn = DATUM.
D Schaltende = DATUM.

D Navigationdaten = (‚Rubrikeinträge‘) + (‚Unterrubrikeinträge‘) +
(‚Stadteinträge‘) + (‚Kundeneinträge‘).
D ‚Rubrikeinträge‘ = {Rubrikeintrag}.
D Rubrikeintrag = Rubrikname + Rubriknummer.
D Rubrikname = STRING.
D ‚Unterrubrikeinträge‘ = {Unterrubrikeintrag}.
D Unterrubrikeintrag = Unterrubrikname + Unterrubriknummer.
D Unterrubrikname = STRING.
D ‚Stadteinträge‘ = {Stadteintrag}.

D Stadteintrag = Stadtname + Stadtnummer.
D Stadtname = STRING.
D ‚Kundeneinträge' = {Kundeneintrag}.
D Kundeneintrag = Kundenname + Kundennummer.
D Kundenname = STRING.

D Schaltdaten_aS = Schaltdaten.

V Sitzungen = Sitzung_Zugangsnummer +
Sitzung_Sitzungsnummer +
'Sitzung_Zeit letzter Aktion' +
Sitzung_Auftragsnummer.
V Sitzung_Zugangsnummer = ZAHL.
V Sitzung_Sitzungsnummer = ZAHL.
V 'Sitzung_Zeit letzter Aktion' = DATE.
V Sitzung_Auftragsnummer = ZAHL.

D Sitzungsnummer = ZAHL.

D Statistik = 'Anzahl Neuaufträge' + 'Anzahl Veränderungsaufträge' + 'Anzahl Verlängerungsaufträge' + 'Anzahl abgelaufener Anzeigen'.
D 'Anzahl Neuaufträge' = "".
D 'Anzahl Veränderungsaufträge' = "".
D 'Anzahl Verlängerungsaufträge' = "".
D 'Anzahl abgelaufener Anzeigen' = "".

D Systemmitteilung = [Zugangsverweigerung | Bearbeitungsabbruch |
Auftragsablehnung | Dateiablehnung | ‚Löschablehnung' |
Eingabeberichtigung].
D Zugangsverweigerung = TEXT.
D Bearbeitungsabbruch = TEXT.
D Auftragsablehnung = TEXT.
D Dateiablehnung = TEXT.
D ‚Löschablehnung' = TEXT.
D Eingabeberichtigung = TEXT.

D Zugangsdaten = Zugangsnummer + Passwort + Benutzerart.
D Zugangsnummer = ZAHL.
D Passwort = STRING.
D Benutzerart = ZAHL.

Anhang D – Logisches Schema

Das vorgestellte logische Schema ist aufgrund der Ausführungen in Abschnitt II.3.1.3 zugleich das Datenbankschema des ASV-Systems.

LOGIC SCHEMA

Name:
ASV-System

Relationsschemata:
Auftrag, Anzeige, Benutzer, Sitzung, Rubrik, Unterrubrik, Stadt,
geschaltet_Anzeige, geschaltet_Auftrag

Integrity Constraints:
Auftrag[Kundennummer] ⊆ Benutzer[Zugangsnummer]
Auftrag[Anzeigennummer] ⊆Anzeige[Anzeigennummer]
Auftrag[Rubriknummer] ⊆ Rubrik[Rubriknummer]
Auftrag[Unterrubriknummer] ⊆ Unterrubrik[Unterrubriknummer]
Auftrag[Verwalternummer] ⊆ Benutzer[Zugangsnummer]
Anzeige[Kundennummer] ⊆ Benutzer[Zugangsnummer]
Anzeige[Rubriknummer] ⊆ Rubrik[Rubriknummer]
Anzeige[Unterrubriknummer] ⊆ Unterrubrik[Unterrubriknummer]
Sitzung[Zugangsnummer] ⊆ Benutzer[Zugangsnummer]
Sitzung[Auftragsnummer] ⊆ Auftrag[Auftragsnummer]
Unterrubrik[Rubriknummer] ⊆ Rubrik[Rubriknummer]
geschaltet_Anzeige[Stadtnummer] ⊆ Stadt[Stadtnummer]
geschaltet_Anzeige[Anzeigennummer] ⊆ Anzeige[Anzeigennummer]
geschaltet_Auftrag[Stadtnummer] ⊆ Stadt[Stadtnummer]
geschaltet_Auftrag[Auftragsnummer] ⊆ Auftrag[Auftragsnummer]

weitere:
Für eine Anzeige (Anzeigennummer) dürfen höchstens ein Veränderungsauftrag und höchstens ein Verlängerungsauftrag existieren;

RELATIONSSCHEMATA

Name:	Auftrag		
Attribute:	Auftragsnummer:	integer	not null
	Neuauftrag:	char(1)	
	Verlaengerungsauftrag:	char(1)	
	Veraenderungsauftrag:	char(1)	
	Kundennummer:	integer	not null
	Anzeigennummer:	integer	
	abgeschlossen:	char(1)	
	Auftragsdatum:	date	not null

	StartHTMLSeite:	charakter(30)
	Ueberschrift:	charakter(80)
	Schlagwoerter:	charakter(100)
	Kurzbeschreibung:	charakter(255)
	KontaktEmail:	charakter(80)
	Rubriknummer:	integer
	Unterrubriknummer:	integer
	Schaltbeginn:	date
	Schaltende:	date
	Verlaengerungsende:	date
	Verwalternummer:	integer
Key:	Auftragsnummer	
FD:	[Unterrubriknummer] -> [Rubriknummer]	

Notes:
Ein Auftrag muß entweder ein Neuauftrag oder ein Veränderungsauftrag oder ein Verlängerungsauftrag sein.
Zulässige Einträge in den Attributen *Neuauftrag*, *Veraenderungsauftrag*, *Verlaengerungsauftrag* und *abgeschlossen* sind ‚0' und ‚1'.
Das Datum für das *Schaltende* muß nach dem Datum für *Schaltbeginn* liegen, das Datum für *Schaltbeginn* muß größer oder gleich dem *Auftragsdatum* sein, das Datum für *Verlaengerungsende* muß nach dem *Auftragsdatum* liegen oder *null* sein.
Attribute *StartHTMLSeite*, *Ueberschrift*, *KontaktEmail*, *Rubriknummer*, *Schaltbeginn* und *Schaltende* müssen *not null* sein, wenn *abgeschlossen* auf ‚1' gesetzt wird.
Attribut *Anzeigennummer* wird aufgrund der Festlegung im Pseudocode, Funktion *Lege neuen Auftrag an*, nicht als *not null* vereinbart, darf aber nur bei Neuaufträgen *null* sein.

Name:	Anzeige		
Attribute:	Anzeigennummer:	integer	not null
	Kundennummer:	integer	not null
	StartHTMLSeite:	charakter(30)	not null
	Ueberschrift:	charakter(80)	not null
	Schlagwoerter:	charakter(100)	
	Kurzbeschreibung:	charakter(255)	
	KontaktEmail:	charakter(80)	not null
	Rubriknummer:	integer	not null
	Unterrubriknummer:	integer	
	Schaltbeginn:	date	not null
	Schaltende:	date	not null
	ausgesetzt:	char(1)	
	abgelaufen:	char(1)	
Key:	Anzeigennummer		
FD:	[Unterrubriknummer] -> [Rubriknummer]		

Notes:
Zulässige Einträge in den Attributen *ausgesetzt* und *abgelaufen* sind ‚0' und ‚1'.
Das Datum für das *Schaltende* muß nach dem Datum für *Schaltbeginn* liegen.

Name:	Benutzer		
Attribute:	Zugangsnummer:	integer	not null
	Benutzerpasswort:	charakter(20)	not null
	Benutzername:	charakter(20)	not null
	Benutzerart:	charakter(10)	not null
Key:	Zugangsnummer		
FDs:	-		

Notes:
Zulässige Einträge im Attribut *Benutzerart* sind: ‚Verwalter' oder ‚Kunde'.

Name:	Sitzung		
Attribute:	Sitzungsnummer	integer	not null
	Zugangsnummer:	integer	not null
	Zeit_letzter_Aktion:	datetime	not null
	Auftragsnummer:	integer	
Key:	Sitzungsnummer		
FDs:	-		

Name:	Rubrik		
Attribute:	Rubriknummer	integer	not null
	Rubrikname:	charakter(30)	not null
Key:	Rubriknummer		
FDs:	-		

Name:	Unterrubrik		
Attribute:	Unterrubriknummer	integer	not null
	Rubriknummer	integer	not null
	Unterrubrikname:	charakter(30)	not null
Key:	Unterrubriknummer		
FDs:	-		

Name:	Stadt		
Attribute:	Stadtnummer	integer	not null
	Stadtname:	charakter(30)	not null
Key:	Stadtnummer		
FDs:	-		

Name:	geschaltet_Anzeige		
Attribute:	Stadtnummer	integer	not null
	Anzeigennummer:	integer	not null
Key:	Stadtnummer, Anzeigennummer		
FDs:	-		

Name:	geschaltet_Auftrag		
Attribute:	Stadtnummer	integer	not null
	Auftragsnummer:	integer	not null
Key:	Stadtnummer, Auftragsnummer		
FDs:	-		

Erklärung

Ich versichere, daß ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Leipzig, August 1999