Universität Leipzig

Fakultät für Mathematik und Informatik
Institut für Informatik

Vergleich von javabasierten Application-Servern auf Basis der Entwicklung eines verteilten Beschaffungssystems für die Universität Leipzig

Diplomarbeit

Aufgabenstellung und Betreuung:
Prof. Dr. habil E. Rahm
Dipl. Math. R. Müller
Dipl. Inf. U. Greiner

vorgelegt von
Lars Quiring

Leipzig, Januar 2003

Vorwort

Die vorliegende Arbeit entstand in Kooperation zwischen dem Institut für Informatik und dem Dezernat 1 der Universität Leipzig. Ursprünglich als Prototyp geplant wurde die entwickelte Anwendung während der Erstellungsphase der vorliegenden Arbeit zu einem produktiven System erweitert, dass seit April 2002 erfolgreich in der Universität Leipzig eingesetzt wird.

Bei folgenden Personen möchte ich mich bedanken:

Herrn Professor Dr. E. Rahm für die Positionierung des Diplomarbeitsthemas in einem Bereich, der Synergien zu meiner beruflichen Orientierung ermöglicht,

Herrn Dr. U. Löser für die Unterstützung bei der Durchsetzung von Änderungen in den Arbeitsprozessen des Dezernats 1,

Herrn G. Franke für seine inhaltliche Unterstützung bei der Konzeptionierung der erstellten Anwendung und der erfolgreichen Einführung als Produktionssystem,

Herrn R. Müller für die intensive Betreuung der Arbeit,

Frau U. Greiner, ebenfalls für die intensive Betreuung der Arbeit.

Herr G. Franke verstarb am 20. Oktober 2002 unerwartet. Ich bedaure seinen Tod.
Er war ein jederzeit offener und fairer Gesprächspartner und maßgeblich für die erfolgreiche Einführung der Beschaffungsanwendung verantwortlich.

Inhaltsverzeichnis

1 Einleitung

1.1 Übersicht

Die vorliegende Arbeit entstand im Rahmen einer Kooperation zwischen dem Institut für Informatik und dem Dezernat 1, Sachgebiet 12 der Universität Leipzig. Ziel der Kooperation ist die rechnergestützte Durchführung von Bestellprozessen zwischen den Einrichtungen (Fakultäten, Institute und Dezernate) und der zentralen Beschaffung. Die Unterstützung der Bestellprozesse verspricht eine deutliche Verbesserung in der Kommunikation zwischen den Einrichtungen und der zentralen Beschaffung und führt somit zu einer messbaren Realisierung von Einsparpotentialen auf beiden Seiten sowie zu einer Qualitätssteigerung der durch das Sachgebiet 12 angebotenen Dienstleistungen.

Die im Kapitel 2 analysierten Bestellprozesse konzentrieren sich auf den Teilbereich der Beschaffung, der sich mit dem Verkauf von Büro- und Bewirtschaftungsmaterial an die Fakultäten, Institute und Dezernate der Universität befasst. Aufgabe der Applikation ist neben der Unterstützung der Bestellprozesse auch die Verwaltung des Lagers, um die vorhandene und veraltete Lagersoftware ersetzen zu können.

Als Framework für die Anwendung wurde eine *javabasierte Mehrschicht-Architektur* gewählt. Die Entscheidung für diese Architektur wird in Kapitel 3 im Rahmen der *Konzeption* der entwickelten Anwendung erläutert. Sie ergibt sich einerseits aus den vom Sachgebiet 12 definierten Zielen und den sich aus der Analyse ergebenden Forderungen, andererseits aus der Flexibilität dieses Ansatzes, insbesondere im Hinblick auf die Portierbarkeit, der klaren Unterscheidung von Nutzerinterface, Business-Logik und Datenbanksystem sowie der konsequenten Umsetzung gut dokumentierter Standards. Der zweite Teil des 3. Kapitels beschreibt detailliert die aus dem Konzept abgeleitete *Implementierung*.

Besondere Aufmerksamkeit widmet die Arbeit im Kapitel 4 der Frage nach Leistungsfähigkeit und Handhabbarkeit der auf Basis der gewählten Architektur eingesetzten *javabasierten Applicationserver*. Hierzu werden anhand der Applikation zwei Produkte bezüglich verschiedener Eigenschaften verglichen. Die in diesem Rahmen durchgeführten Untersuchungen sollen erste Entscheidungskriterien der Wahl einer geeigneten Produktsumgebung für den Fall sein, dass die entwickelte Anwendung unter hohen Lasten betrieben wird. Genauer betrachtet werden dabei das *Verwaltungs- und Administrationsmodell* der Application-Server sowie ihre *Performance* hinsichtlich spezifischer Aufgaben. Insbesondere wird die *Verwaltung von Session-Daten* in den Blickpunkt gerückt. Sie ist eine wesentliche Fähigkeit der Application-Server und wird mit Hilfe eines erweiterten Codesegments aus der entwickelten Anwendung untersucht.

Das Kapitel 5 wirft ein Schlaglicht auf die Einsatzmöglichkeit eines *Data Warehouse* im Rahmen der entwickelten Anwendung. Aus den im produktiven Betrieb gewonnenen Erfahrungen über häufig angeforderte und zeitaufwendige Auswertungen werden Anforderungen an ein *Data Warehouse* abgeleitet. Der darauf aufbauende Entwurf kann als Beispiel der Integration dieser Technologie in e-Procurement – Systeme wie auch als zukünftige Entwicklungsrichtung der vorgestellten Beschaffungsanwendung verstanden werden. Im folgenden Kapitel schließt sich eine kritische Betrachtung der entwickelten Anwendung, eine zusammenfassende Einschätzung der verglichenen Applicationserver und ein Ausblick auf die sich durch die Nutzung eines *Data Warehouses* ergebenden Möglichkeiten für denkbare weiterführende Entwicklungsstufen der Anwendung an.

Die im Rahmen der Diplomarbeit entwickelte Applikation befindet sich seit April 2002 im produktiven Betrieb. In den ersten sechs Monaten wurden von den etwa 300 universitätsinternen Nutzern mehr als 8000 Bestellungen über das System abgewickelt.

1.2 Ausgangssituation und Zielstellung des Dezernats 1

1.2.1 Aufgabenbereich der Beschaffung im Dezernat 1

Die Instanz für die Beschaffung von Artikeln aus zentralbewirtschafteten Titeln[1] der Universität Leipzig ist das Sachgebiet 12 im Dezernat 1 der Universität Leipzig. Seine Hauptaufgabe ist das Bedienen von jährlich etwa 12000 Bestellungen des Büro- und Bewirtschaftungsbedarfs durch die Fakultäten, Institute und Dezernate der Universität. Das Sachgebiet 12 verfügt hierzu über ein eigenes Lager, in dem etwa 85% der regelmäßig bestellten Artikel vorrätig sind. Der Bestand beinhaltet größtenteils Güter, deren Wert pro Stück gering ist, die aber in großer Menge verbraucht werden.
Aus dieser Aufgabe erwachsen die folgenden vier Tätigkeitsfelder für die Mitarbeiter des Sachgebiets 12:

1. Vorbereitung von Ausschreibungen, die Verhandlung von Rahmenverträgen mit den Lieferanten und das Auslösen von Bestellungen bei denselben.
2. Verwaltung des Artikellagers
3. Bearbeitung der von den Fakultäten, Instituten und Dezernaten[2] ausgelösten Bestellungen und Bereitstellung der bestellten Artikel
4. Umbuchung der bestellten Warenwerte durch Entlastung der Kostenstelle des Lagers und Belastung der Kostenstelle der bestellenden Einrichtung

[1] Titel sind die Zuordnungsbereiche der Haushaltsmittel der Universität.

[2] Im Folgenden werden Fakultäten, Institute und Dezernate auch unter dem Begriff Einrichtungen zusammengefaßt

Die unter Punkt 1 genannten Tätigkeiten betreffen die Kommunikation des Sachgebiets 12 mit externen Lieferanten. Insbesondere Ausschreibungen und die Verhandlung der Rahmenverträge erfordern eine weitgehend individuelle Bearbeitung und sind deshalb nicht Gegenstand der folgenden Ausführungen. Dahingegen eröffnen sich bei näherer Betrachtung der in den drei Folgepunkten beschriebenen Abläufe eine Reihe von Möglichkeiten für eine softwaregestützte Optimierung. Die nachstehende Abbildung 1-1 verdeutlicht im Überblick, in welchen Rollen Mitarbeiter des Sachgebiets 12 und der Einrichtungen in den genannten Tätigkeitsbereichen agieren. Die Aufschlüsselung der Prozessschritte, die zwischen den Mitarbeitern in ihren verschiedenen Rollen ablaufen, wird in Kapitel 2 unternommen.

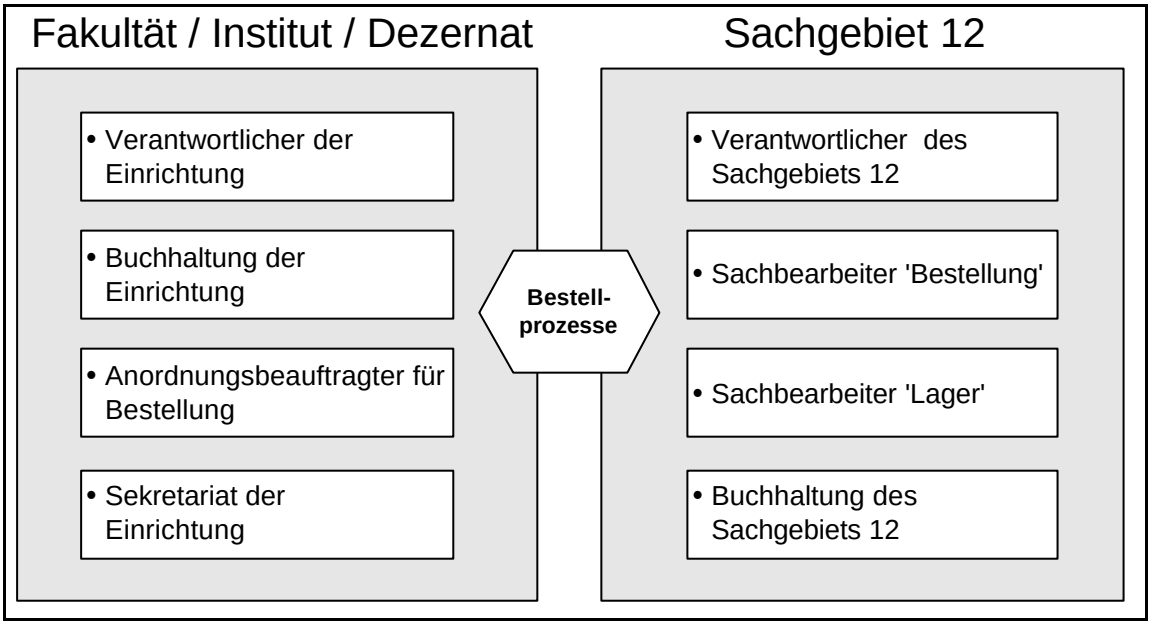


Abbildung 1-1: Akteure der Bearbeitung einer Materialbestellung. An der Beschaffung von Material sind Mitarbeiter in acht verschiedenen Rollen beteiligt. Durch Personalunion kann die Anzahl der beteiligten Personen kleiner sein.

Aus der Betrachtung der genannten Tätigkeitsfelder ermittelte das Sachgebiet 12 unter der Maßgabe einer Verbesserung der allgemeinen Arbeitseffizienz die im folgenden Abschnitt genannten Ziele.

1.2.2 Zielstellung

Um die Zielstellung für die Entwicklung einer Softwarelösung zur Unterstützung der Beschaffungsprozesse definieren zu können, wurden die Prozessketten der vorstehend genannten Tätigkeitsfelder genauer betrachtet und folgende Ziele im Hinblick auf eine effizientere Gestaltung des Bestell- und Lagerwesens ermittelt:

A. Optimierung der Prozessschritte in der Kommunikation zwischen Fakultäten, Instituten und Dezernaten
B. Entwicklung einer Software, mit deren Hilfe Fakultäten, Dezernate und Institute alle Bestellungen und bestellnahen Aktivitäten (Einsicht alter Bestellungen, Listendruck, etc.) softwareunterstützt abwickeln können
C. Entwicklung einer Software, welche das Sachgebiet 12 bei der Bereitstellung und Lagerverwaltung unterstützt (Austausch des bisherigen Lagerprogramms durch ein anderes Produkt, um die Pflege und Weiterentwicklung zu sichern)
D. Erfüllung der steigenden Anforderungen zur Analyse des Mitteleinsatzes und der Bereitstellung von Daten für die Kosten/Leistungsrechnung der Universität
E. Kompatibilität des ab 2002 eingeführten Kostenartenrahmens, welcher der Gruppierung und eindeutigen Identifikation aller bestellbaren Artikel dient
F. Anbindungsmöglichkeit der zu entwickelnden Software an das von der Universität zur internen Verwaltung der Mittel eingesetzte Programm HIS MBS[3]
G. Geringe Anforderungen an die Rechnerausrüstung der etwa 350 Besteller aus Fakultäten, Dezernaten und Instituten. Einfachheit der Handhabung, da die Unterstützung bei Betriebsproblemen auf Seiten der Besteller nicht gewährleistet werden kann
H. Offenheit gegenüber Erweiterungen und Anbindungen von bestehenden wie auch neuen Applikationen

Das bisher im Bereich der Lagerwirtschaft eingesetzte Programm *HIS BEL*[4] konnte die genannten Zielsetzungen nicht befriedigend vorantreiben, da die Software einerseits nicht den geforderten Funktionsumfang besaß, ihre Weiterentwicklung andererseits offiziell eingestellt wurde. In der zweiten Jahreshälfte 2000 bestand akuter Handlungsbedarf, weil eine weitere Pflege der Daten in HIS BEL durch die von dem Programm nicht darstellbare Milleniumumstellung unmöglich wurde.
Das Dezernat 1 entschied deshalb, in Zusammenarbeit mit dem Institut für Informatik eine *Bestandsaufnahme der Prozesse* innerhalb der Materialbeschaffung durchzuführen und sie zu optimieren. Auf dieser Ausgangsbasis sollte ein Softwaresystem konzeptionell entwickelt und implementiert werden, um die genannten Vorgänge im Rahmen der Tätigkeiten des Sachgebiet 12 zu unterstützen.
Die vorstehend dargestellten Überlegungen zur Optimierung von Bestellprozessen fallen in den Bereich des sogenannten *e-Procurements*. Der folgenden Abschnitt dient der Klärung dieser

[3] HIS MBS ist eine von der Firma HIS entwickelte Software zur Mittelbewirtschaftung und wird bundesweit in Universitäten eingesetzt. Im weiteren Verlauf wird auch das Programm HIS BEL genannt. Es dient der Unterstützung der Lagerverwaltung und dem Bestellwesen und ist von HIS MBS zu unterscheiden.

[4] Detaillierte Informationen zur HIS GmbH und dem Produkt HIS BEL liefert der Abschnitt 2.3..

Begrifflichkeit und versucht eine Einordnung der Ansätze des Sachgebiets 12 aus betriebswirtschaftlicher Sicht.

1.2.3 e-Procurement

Die Summe der ökonomischen und technischen Lösungen wird seit einigen Jahren unter dem Stichwort *e-Procurement* zusammengefasst: Hiermit ist der Teilbereich an Geschäftsbeziehungen gemeint, die sich zwischen Lieferanten und einkaufenden Unternehmen abspielen (vgl. [BN96]). Obwohl es die Bezeichnung e-Procurement erst seit zwei Jahren gibt, wird gerade diesem Bereich ein besonders hohes Potential zugesprochen (vgl. [SN99])[5].

Beschaffungsvorgänge sind in großen Unternehmen insgesamt von einer hohen Heterogenität gekennzeichnet: Es gibt eine Vielzahl von zu beschaffenden Gütern, sowie von Akteuren im Unternehmen, die mit der Beschaffung beschäftigt sind, und auch von Quellen für die Beschaffung (vgl. [EY99]).

1.2.3.1 Systematisierung von Einsparungspotentialen

Eine kostenspezifische Effizienzanalyse setzt an dieser Konstellation einer Vielzahl von Beziehungen an. Folgt man der betriebswirtschaftlichen Literatur [BA99], so lassen sich mit einem e-Procurement-Ansatz *Kostenreduktionspotentiale* einerseits und *Umsatzsteigerungspotentiale* anderseits nutzen. Mit Potentialen zur Umsatzsteigerung ist in erster Linie der Aufbau eines neuen Geschäfts im B2B-Bereich[6] gemeint. Ein Unternehmen bietet seine Produkte auf einem elektronischen Marktplatz an, den potentielle Kunden für ihren Einkauf nutzen können (vgl. zu Marktmachtkonstellationen elektronischer Marktplätze [TA99], S.59ff.). Für einige Unternehmen bedeutet dieser Schritt den Markteintritt in den als zukunftsträchtig geltenden B2B-Bereich.

In der vorliegenden Arbeit sind jedoch die Möglichkeiten einer Kostenreduktion von primärem Interesse. Kostenreduktionen werden in erster Linie in den Bereichen (a) Prozesskosten und Materialkosten (b) des Einkaufs gesehen. Diese Unterscheidung ist wichtig, denn bei jedem zu beschaffenden Gut sind die Kostenanteile äußerst unterschiedlich: Die Kosten für den Einkauf der Wertschöpfungsstufe Druck (Druckkosten ohne Papier) eines Buches im Format "Standard-Schwarz-Weiß" verteilen sich beispielsweise zu 95% auf die Materialkosten. Der Prozess des Einkaufs hingegen zeichnet für lediglich 5% der Kosten verantwortlich. Anders hingegen ist der Kostenanteil bei einem Bleistift. Hier schlagen die Materialkosten mit nur 3% oder weniger, die Prozesskosten hingegen mit 97% zu Buche.

---

[5] Ein weiterer Indikator könnte der globale Umsatz von B2B-Software sein, zu dem E-Procurement-Software 78% beiträgt (vgl. Datamonitor, zit. nach [GC00], S. 7).

[6] Die Abkürzung B2B bezeichnet den Teilbereich der wirtschaftlichen Beziehungen, die zwischen Unternehmen existieren. Analog dazu stehen die Abkürzungen B2C für die Beziehung von Unternehmen zu Privatkunden.

1.2.3.2 Kostensenkungspotentiale bei Beschaffungsvorgängen der Universität Leipzig

Zur Erklärung der Realisierung der Kostenvorteile ist es zweckmäßig, den Einkaufsprozess als Bezugspunkt zu wählen. Es lassen sich drei Phasen des Einkaufsprozesses unterscheiden, die ein e-Procurement-System unterstützen kann ([NG99], S. 290): Der Prozess umfasst die *Informationssammlung*, die *Preisbildung* sowie die *Abwicklung* der Beschaffung an sich.

Bei der *Informationssammlung* geht es um Informationen über die einzukaufenden Produkte oder Dienstleistungen und um Informationen über die Lieferanten. Ein e-Procurement kann hier beispielsweise durch ein umfangreiches Katalogsystem das schnelle Auffinden und Auswählen von Produkten erleichtern.

In der Phase der *Preisbildung* soll thematisiert werden, auf welche Arten es zu einem Einkaufspreis kommen kann. Hier kann ein e-Procurement durch Rahmenverträge oder Katalogpreise die Preisbildung überflüssig machen bzw. durch ein Auktionssystem eine systematische Preisbildung betreiben.

Die Phase der *Abwicklung* schließlich meint die operative Abwicklung des Einkaufsgeschäfts vom Eingang der Bestellung beim Lieferanten bis zur Bezahlung. Ein e-Procurement erleichtert die Abwicklung insbesondere durch eine standardisierte und automatisierte Datenübermittlung.

Die folgende Abbildung veranschaulicht diese drei Phasen und die Unterstützungsmöglichkeit durch ein e-Procurement.

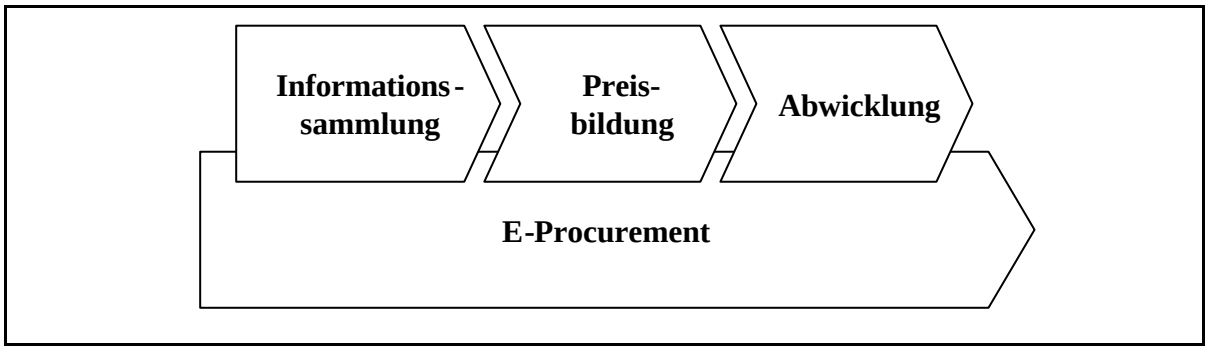


Abbildung 1-2: Phasen des Einkaufsprozesses und e-Procurement (Quelle: nach [NG 99], S. 290)

Projiziert man die drei Phasen auf die vom Sachgebiet 12 definierten Ziele, ergeben sich die Schwerpunkte der entwickelten Applikation. Sie liegen im Bereich der *Informationssammlung* und der *Abwicklung*. Die Informationssammlung ermöglicht durch eine strukturierte, detaillierte und aktuelle Darstellung der angebotenen Artikel, ihrer Preise und Verfügbarkeit eine deutliche Verkürzung der Prozessketten und reduziert die Wahrscheinlichkeit von Reklamationen. Gleichzeitig ist sie die Voraussetzung für die Preisbildung und Abwicklung. Die Preisbildung hat im Fall der zentralen Beschaffung eine geringe Relevanz für die Prozesskostenoptimierung, weil die Einrichtungen nicht mit der zentralen Beschaffung um Preise verhandeln. Entweder sie beziehen die Artikel zu dem von

der zentralen Beschaffung in Rahmenverträgen definierten Preis, oder sie wenden sich direkt an eigene Lieferanten.
Für die *Abwicklung* jedoch ergeben sich aus der strukturierten Informationssammlung große Einsparpotentiale. Die im Kapitel 2 analysierten Prozesse zur Abwicklung eines Artikelkaufs von der Bestellung bis zur Bestätigung des Warenerhalts sind aufwendig und werden in der entwickelten Applikation in optimierter Form weitgehend unterstützt.

1.3 Das Java-Framework als Basis der Beschaffungsanwendung

Die im vorstehenden Kapitel aus Sicht des e-Procurement kategorisierten Beschaffungsprozesse sollen von der entwickelten Software unter Erfüllung der im Abschnitt 1.2.2 genannten Ziele unterstützt werden. Insbesondere die Forderungen nach einer beim Nutzer möglichst wartungsfrei funktionierenden Applikation, nach Offenheit gegenüber neuen Entwicklungen durch die konsequente Umsetzung von offenen Standards und der damit einhergehenden Forderung nach der Skalierbarkeit der Anwendung favorisieren eine Drei-Schicht-Architektur. Im Bereich der Drei-Schicht-Architekturen präsentieren sich auf Java basierende Application-Server mit einem hoch entwickelten Reifegrad und einer sehr guten Dokumentationslage [ST02]. Dies ist wichtig für die im Rahmen der vorliegenden Arbeit ebenfalls durchgeführten Untersuchungen zum Laufzeitverhalten verschiedener Produktausprägungen von Application-Servern. Somit wurde das von Sun Microsystems entwickelte *Java 2 Enterprise Edition Framework* als Basis der erstellten Beschaffungsanwendung gewählt. Die folgenden Abschnitte stellen das Framework in einer kurzen Übersicht dar und positionieren die erstellte Anwendung anhand der genutzten Komponenten.

1.3.1 Die verschiedenen Editionen von Java

Die Programmiersprache Java wurde von Sun Microsystems seit der Version 1.2 in drei Ausprägungen standardisiert:

- Micro Edition J2ME
- Standard Edition J2SE
- Enterprise Edition J2EE

Die Micro Edition der Java 2 Plattform beinhaltet eine stark auf Kleinstgeräte wie PDAs (Personal Digital Assistent) optimierte und reduzierte Version der Java Virtual Machine. Sie soll vor allen Dingen auch für Kleinstgeräte mit wenig Speicher und Prozessorleistung und stark unterschiedlichen technischen Gegebenheiten eine weitgehend plattformunabhängige Entwicklung von Programmen

ermöglichen. Selbst manche Handys können mittlerweile auf diese Weise mit Programmen ausgestattet werden.
Die Standard Edition der Java 2 Plattform beinhaltet alle für die Applikationsentwicklung notwendigen Komponenten, setzt jedoch einige durch den J2EE-Standard festgelegten Eigenschaften nicht um wie bspw. die Unterstützung von Enterprise Java Beans.
Die als letzter Enumerationspunkt genannte Java 2 Enterprise Edition definiert einen Standard zur Entwicklung von Mehrschichten-Anwendungen. J2EE-basierte Applikationen setzen auf dieser Plattform auf und werden modular, komponentenbasiert und in Richtung eines festen Standards - *der J2EE-Spezifikation* - entwickelt [SS02]. Damit wird eine Portierung mit minimalem Aufwand auf jeden J2EE-konformen Application Server gewährleistet.

1.3.2 Die Beschaffungsanwendung aus Sicht des J2EE–Design- und Programmiermodells

Das J2EE – Design- und Programmiermodell stellt ein Konzept für eine komponentenbasierte, modulare, schichten-orientierte und verteilte Softwareentwicklung dar, welches aus mehreren deutlich unterschiedenen, logischen Schichten besteht. J2EE-Applikationen können auf physisch verschiedenen Rechner-Systemen installiert werden. Obwohl J2EE-Applikationen über drei, vier oder mehr Schichten verteilt sein können, wird generell von einer *Drei-Schicht-Architektur* ausgegangen, die in der folgenden Abbildung schematisch dargestellt wird:

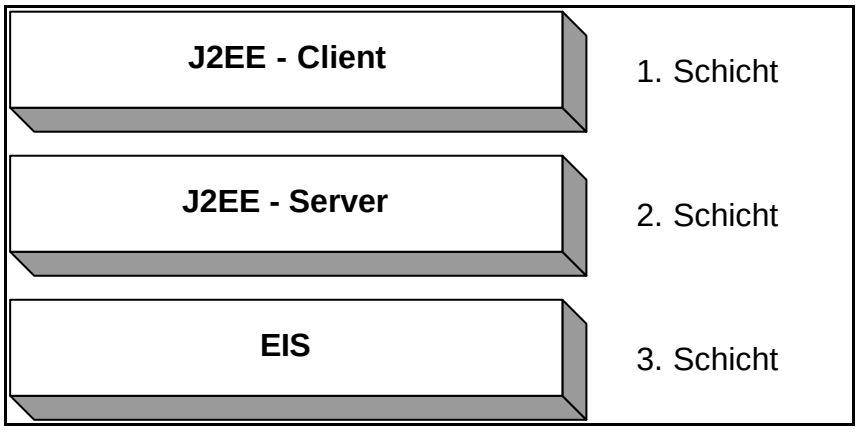


Abbildung 1-3: Drei-Schicht-Architektur im J2EE Design- und Programmiermodell mit J2EE-Client, -Server und Enterprise Information System [SS02]

1.3.2.1 Schicht 1: Der J2EE-Client

Der J2EE-Client bildet die *Präsentationsschicht.* Er hat die Funktion eines Benutzer-Frontends[7] und kann in der Ausprägung eines Web-Clients oder Applikations-Clients entwickelt werden.

[7] Das Benutzer-Frontend stellt dem Anwender die Funktionalitäten eines Programms in einer für ihn lesbaren und nutzbaren Form zur Verfügung.

Web-Clients bestehen aus zwei Teilen: dynamischen Web-Seiten, welche von den sogenannten Web-Komponenten in der 2. Schicht generiert werden, und einem Web-Browser, der die Seiten interpretiert und darstellt, welche zuvor vom Server empfangen wurden.
Applikations-Clients besitzen üblicherweise eine graphische Benutzeroberfläche, meist realisiert auf Basis von Swing- oder AWT- (Abstract Windowing Toolkit) Klassen[8] der Java-Ungebung des Clients.

Im Rahmen der für das Sachgebiet 12 erstellten Anwendung wurde als J2EE-Client ein Web-Client entwickelt, um die im Abschnitt 1.2.2 unter dem Enumerationspunkt G aufgeführten Forderungen nach

1. einer geringen notwendigen Rechnerausstattung ,
2. einfacher Handhabbarkeit und
3. einem möglichst geringen Aufwand bei der Sicherung der Betriebsbereitschaft des Nutzers erfüllen zu können.

1.3.2.2 Schicht 2: Der J2EE-Server

In der mittleren Schicht liegt der *Application-Server* als Laufzeitumgebung für die Applikation. Er bildet die *Laufzeitumgebung für die Komponenten der J2EE-Applikation* und den Kern der J2EE-Plattform. Der Application-Server enthält die Geschäftslogik der Anwendungen sowie einen Ressourcenmanager. Dabei fungiert er als Server gegenüber dem J2EE-Client, aber auch als Client gegenüber der dritten Schicht, bspw. einem Datenbankserver.
Die Application-Server - Umgebung kann, wie in Abbildung 1-4 ersichtlich ist, grundsätzlich in zwei verschiedene, sogenannte Engines mit den dazugehörigen Diensten unterteilt werden:

Die *Servlet-Engine*, auch Web-Container genannt, hat die Verwaltung von Java Servlets, JavaServer Pages und JavaBeans zur Aufgabe. Voraussetzung ist serverseitig die Installation einer Java VM (Virtual Machine), der Java-Laufzeitumgebung. Die Servlet-Engine ist somit ein Programm, welches auf einer solchen VM als Basis läuft und Servlets ausführen kann.

[8] Swing- und AWT-Klassen bieten dem Programmierer jeweils einen Satz von Funktionalitäten an, welche ihm die Nutzung von typischen Elementen graphischer Anwendungsoberflächen wie Buttons, Drag-and-Drop, etc. erleichtern..

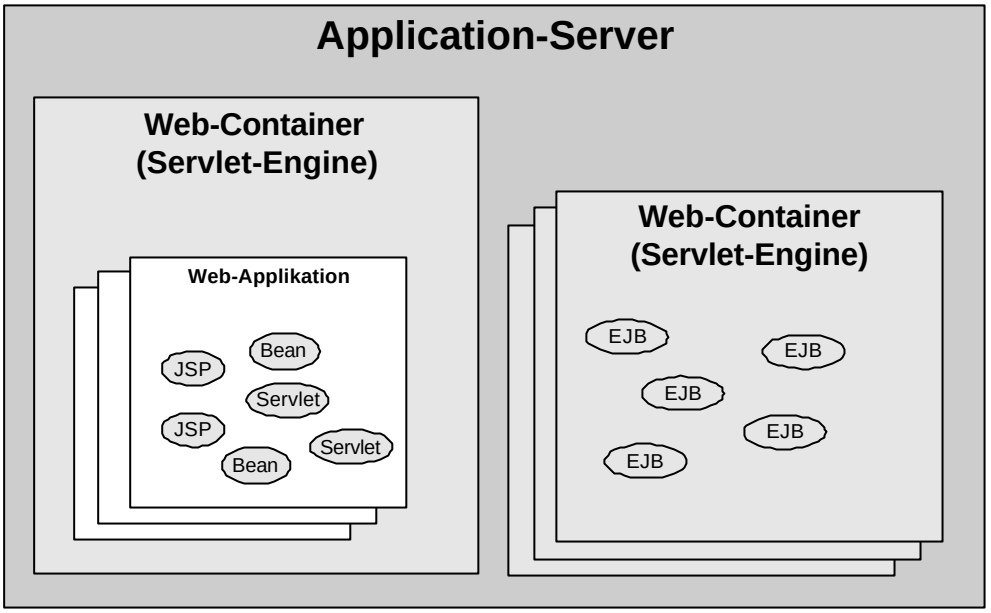


Abbildung 1-4: Containertypen in javabasierten Application Servern.

EJB-Container haben die Aufgabe, Enterprise JavaBeans zu verwalten und auszuführen. Er stellt eine logische Abstraktion der Dienste eines Application-Servers dar und bietet sie den EJB an.

Im Rahmen der erstellten Anwendung wird ausschließlich die *Servlet-Engine* genutzt. Die Anwendung wurde mit JavaServer Pages, JavaBeans und Servlets realisiert. Es wurden keine Enterprise JavaBeans eingesetzt. Ihr Einsatz ist vor allen Dingen dann zu empfehlen, wenn eine Kapselung der Businesslogik notwendig ist, weil mehrere verschiedene Applikationen auf sie zugreifen wollen. Diese Forderung war im Rahmen dieser Arbeit nicht gegeben. Hinzu kommt ein höherer Administrations- und Konfigurationsaufwand sowie die zu Beginn des Projekts fehlende Verfügbarkeit von kostenlosen Production-Releases der Produkte, die einen EJB-Container anboten.

1.3.2.3 Schicht 3: Das Enterprise Information System

Die dritte und unterste Schicht, in der J2EE-Spezifikation Enterprise Information System (EIS) - Schicht genannt, beinhaltet Unternehmens-Infrastruktursysteme wie zum Beispiel Enterprise Resource Planning (ERP) und Datenbank-Systeme. [TP00]

Im Rahmen der entwickelten Applikation wird der *Microsoft SQL Server 2000 Enterprise Edition* als Datenbanksystem eingesetzt. Er stellt in einer leicht erlernbaren Administrationsoberfläche viele notwendige Funktionalitäten eines Datenbanksystems zur Verfügung. Der Hauptgrund für seinen Einsatz ist die Integration der Funktionalitäten, die zum Aufbau eines *Data Warehouses* notwendig sind, dessen Konzept im Kapitel 5 erarbeitet wird.

1.4 Aufgabenstellung der Diplomarbeit

Die vom Dezernat 1 definierten Zielstellungen für den Einsatz einer Softwarelösung im Bereich der Materialbeschaffung erfordern eine *intensive Beschäftigung mit den Bestellprozessen* zwischen den Einrichtungen und dem Sachgebiet 12 des Dezernats. Da noch keine Analyse der Prozesse durchgeführt wurde, ist es notwendig, die verschiedenen in den Beschaffungsvorgang involvierten Akteure zu befragen und aus den jeweiligen Tätigkeiten sowie den genutzten Hilfsmitteln wie Formularen und Programmen ein Gesamtbild der Prozessketten aufzubauen. Die Ergebnisse dieser Studie sind die Grundlage für die Entwicklung des *Konzepts der Softwarelösung*, die neben der Definition von Funktionseinheiten auch die Wahl der Basisarchitektur und das Design der Datenbank enthält. Die *Implementierung* ist auf verschiedenen Ebenen zu leisten. Sie umfasst einerseits das Nutzerfrontend, bei deren Entwicklung insbesondere eine hohe Akzeptanz des Nutzers im Fordergrund steht. Maßgeblich hierfür sind ein transparenter Aufbau der Anwendung und die Umsetzung der vom Dezernat 6 der Universität Leipzig definierten Designrichtlinien für universitätsinterne Internetseiten.

Andererseits ist die Applikationslogik zu implementieren sowie die Datenstruktur aufzubauen. Um die Portierbarkeit der Lösung zu gewährleisten, können keine produktspezifischen Klassen der Application-Server genutzt werden, wie sie bspw. im Bereich des Datenbankzugriffs zumindest von einem der genutzten Produkte eingesetzt werden. Auch diese Klassen müssen unter Einsatz der genutzten Java-Version implementiert werden.

Während sich die vorstehend aufgeführten Aufgaben unter dem Überbegriff: *Konzeption und Entwicklung einer Beschaffungsanwendung* zusammenfassen lassen, ist der zweite Themenblock dem *Vergleich von zwei Application-Servern* auf Basis der entwickelten Anwendung gewidmet.

Verglichen werden im Rahmen des zweiten Schwerpunkts der Arbeit Tomcat in der Version 3.2.3 als Referenzimplementierung des J2EE-Frameworks mit dem Websphere Applicationserver der Version 3.5 von IBM. Während es sich bei ersterem um ein kostenloses Produkt handelt, hat IBM eine ganze Reihe von kostenpflichtigen Produkten im Rahmen der Websphere-Familie entwickelt. Hauptaufgabe dieses Schwerpunktes ist es, einen Vergleich ausgewählter Laufzeit- und Konfigurationseigenschaften der beiden Application-Server durchzuführen.

Als ein Ausblick auf weitere Entwicklungsstufen der Beschaffungsanwendung ist die *Konzeptionierung eines Data Warehouse* zu sehen. Dieser dritte Themenblock innerhalb der vorliegenden Arbeit diskutiert die Möglichkeiten, die der Einsatz eines Data Warehouses insbesondere für die zunehmenden Ansprüche aus dem Bereich des Controllings für die Anwendung haben kann und entwickelt das Konzept für eine mögliche Implementierung.

2 Bestandsaufnahme und Analyse der Beschaffungsprozesse

Die Universität Leipzig hat in der 2. Jahreshälfte 2000 beschlossen, den steigenden Anforderungen zur Analyse des Mitteleinsatzes und der Bereitstellung von Daten für die Kosten/Leistungsrechnung sowie dem wachsenden Einsparungsdruck konstruktiv zu begegnen. Anstatt die Qualität der erbrachten Dienstleistungen zu reduzieren, sollten *die Arbeitsprozesse der im Sachgebiet 12 beheimateten Beschaffung von Büro- und Bewirtschaftungsbedarf analysiert und optimiert werden.* Dies sollte Grundlage für den zweiten geplanten Schritt sein, die *Entwicklung einer Individualsoftware*, welche die umgestalteten Arbeitsabläufe abbildet und unterstützt.
Dieses Kapitel beschreibt die vorgefundenen Arbeitsabläufe. Es nennt auch diejenigen Schritte in den Prozessketten, an denen sich Optimierungen in der gegebenen Situation realistisch umsetzen lassen.

2.1 Quantitative Bestandsaufnahme

Die 6 Dezernate, 12 Fakultäten und etwa 25 selbständigen Institute und zentralen Dienststellen (Stand 2002) der Universität Leipzig lösen jährlich etwa 14.500 Bestellungen für Lagermaterial beim Sachgebiet 12 aus. Jede besteht im Mittel aus vier bis fünf Artikelpositionen. Somit ergeben sich jährlich etwa 60.000 umgeschlagene Artikelpositionen in einem Gesamtwert von zirka 1.100.000,- Euro.
Der Artikelkatalog umfasst 900 bis 1000 Positionen. Bis Mitte 2001 wurden die Artikel lose in Gruppen unterteilt. In Vorbereitung der Einführung der Kosten- und Leistungsrechnung wurden diese Artikelgruppen reorganisiert und in 11 Kostenarten überführt.

Kostenart	Bezeichnung
351	Hygienebedarf
352	Materialverbrauch f. Lager, Hausmeister – Allgem. Material
353	Materialverbrauch f. Betriebstechnik / Hausmeister – Werkzeug
411	Geschäftsbedarf – Verbrauchmaterial – allgemein
412	Geschäftsbedarf – Toner, farbbänder, Tintenpatronen, Disketten, EDV-Zubehör, usw. / techn. Bürobedarf
413	Geschäftsbedarf – Vordrucke, Formulare, Stempel
415	Geschäftsbedarf – Druck- und Buchbindekosten universitätsintern (einschl. Kopier-papier Zentrale Vervielf.)
419	Geschäftsbedarf – Sonstiger / Sondermaterial
441	Dienst- und Schutzbekleidung
449	Sonstiger allgem. Materialverbrauch Arbeitsschutzmaterial

Tabelle 1: Kostenarten laut dem Materialkatalog der Universität Leipzig (Stand 1. Quartal 2001)

Die Kostenarten gruppieren sich einerseits in den *Bewirtschaftungsbedarf*, der in den Kostenarten 351, 352 und 353 etwa 20% aller Artikel subsummiert. Die Kostenarten von 411 bis 449 bilden andererseits den *Bürobedarf* und teilen sich die restlichen ca. 800 Artikel.

Mit der Einführung der Kosten- und Leistungsrechnung wurde auch eine sprachliche Änderung durchgesetzt. Der Begriff Wirtschaftsbedarf wird synonym mit dem Begriff Bewirtschaftungsbedarf verwendet. Letzterer stellt die offizielle Sprachregelung dar und setzt sich langsam im Tagesgeschäft durch, während ersterer noch in manchen Formularen, so auch dem in Abbildung 2-1 dargestellten Bestellformular zu finden ist. Analog verhält es sich mit den Begriffen Geschäftsbedarf und Bürobedarf, wobei letzterer die offizielle Sprachregelung ist.

Grundsätzlich müssen für den Wirtschaft- und Geschäftsbedarf zwei verschiedene Bestellformulare ausgefüllt werden. Die ausgefüllten Bestellformulare erreichen die Beschaffung in 90% aller Fälle über die Hauspost, 8% werden gefaxt und 2% persönlich eingereicht, meist mit dem Ziel, die bestellten Artikel sofort mitnehmen zu können. Die Sachgebiet 12 hat Zeiten bekanntgegeben, zu denen Eilabholer bedient werden können.

Anfordernde Stelle (Fakultät/Institut/Dienststelle)

Dienststellen-Kennziffer:

Projektnr. bei Drittmittelvorhaben:

An
Sachgebiet 12 – Beschaffung –
Goethestr. 6 / Ritterstr. 26
Postfach 4316 **Zentrallager**
Tel.: 97 31121/22/23 Fax: 97 31129

Auskunft erteilt/Anlieferung an:

Herr/Frau _____

Telefonnummer _____

Materialanforderung: Blatt 1
(Geschäftsbedarf getrennt vom Wirtschaftsbedarf anfordern)

Gebäude _____ Zimmer _____

von anfordernder Stelle auszufüllen

Lfd. Nr.	Artikel-Nr.	Menge	Gegenstand	Bemerkung
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				

Ort Datum

Unterschrift des Dienststellenleiters bzw. seines Bevollmächtigten

Abbildung 2-1: Das offizielle Bestellformular. Es wird vom Sachgebiet 12 ausgegeben. Einige Fakultäten haben das Formular digital (bspw. in Microsoft Word) aufgebaut, um die Bestellung jeweils ausdrucken zu können. Die Artikel des Wirtschaftsbedarfs müssen unter Nutzung des gleichen Vordrucks getrennt vom Geschäftsbedarf bestellt werden.

Vorrätig hat das Lager 85% der im Katalog befindlichen Artikel. Die Bestellfrequenz der übrigen Artikel ist so gering, dass die Mitarbeiter eine Bestellung direkt an die Lieferanten des Sachgebiets 12 weitergeben. Die Lieferzeiten sind deshalb in den meisten Fällen verzögert.

Das Sachgebiet 12 macht im *Abstand von zwei Jahren Ausschreibungen*, um den pro Artikelkategorie günstigsten Anbieter zu ermitteln. Mit diesen Lieferanten werden Rahmenverträge geschlossen, die Sonderkonditionen bezüglich der Preise und Lieferzeiten garantieren.

In 2000 unterhielt die Beschaffung Rahmenverträge mit vier Lieferanten des Bürobedarfs und drei Lieferanten des Bewirtschaftungsbedarfs. Sie decken jeweils eine Artikelkategorie ab. Im Bürobedarf sind dies der allgemeine Bürobedarf (Marker, Locher, Schnellhefter etc.), Papiererzeugnisse (ohne Kopierpapier, d.h. Ordner, Register, Hefte), technischer Bürobedarf (Toner, Farbpatronen, Disketten, etc.) sowie Kopier- und Druckerpapier. Im Bewirtschaftungsbedarf gibt es jeweils einen Lieferanten für Werkzeuge, Sanitärerzeugnisse und Sonder- wie Elektromaterial. Im Durchschnitt werden bei allen Lieferanten zusammen monatlich etwa 150 Bestellungen ausgelöst.

Darüber hinaus gab es einige Lieferanten, die bei ausgefallenen Artikelbestellungen herangezogen wurden (bspw. Spezialdruckpapier). Etwa 95% der vom Sachgebiet 12 jährlich ausgelösten Bestellungen für Artikel des Büro- und Bewirtschaftungsbedarfs entfallen auf die sieben durch Rahmenverträge gebundenen Lieferanten.

Die Einrichtungen haben die Möglichkeit, auch solche Artikel zu bestellen, die nicht im Katalog verzeichnet sind. Grundsätzlich empfiehlt das Dezernat jedoch, sich an den dort empfohlenen Artikeln zu orientieren, da sie mit Bedacht auf ein vernünftiges Preis-/Leistungsverhältnis ausgewählt wurden. Ausnahmen können auftreten, wenn bspw. naturwissenschaftliche Fakultäten aufgrund Ihrer Forschungen besondere Materialien benötigen. Solche Artikel werden beschafft, jedoch nicht in den Artikelkatalog eingegliedert, da sie mit hoher Wahrscheinlichkeit nicht von anderen Einrichtungen benötigt werden.

2.2 Bestandsaufnahme der vorhandenen Kenntnisse und Ausstattung

Das Spektrum der Kenntnisse der potentiellen Nutzer einer Beschaffungsanwendung ist sehr breit. Gleiches gilt für die vorhandene technische Infrastruktur. Die folgende kurze Bestandsaufnahme liefert Informationen für die Entwicklung des Userinterface im Konzept der Beschaffungsanwendung. Sie ergab sich im Rahmen der durch das Dezernat 1 gestellten Anfrage an alle Einrichtungen, die bestellberechtigten Personen namentlich aufzuführen, um sie in Beschaffungsanwendung als Nutzer eintragen zu können. Etwa 250 Mitarbeiter der Universität Leipzig haben die Berechtigung, auf eine oder mehrere Kostenstellen Artikel des Büro- und Bewirtschaftungsbedarfs zu bestellen.

Infrastruktur

Das Gros der möglichen Nutzer einer Beschaffungsanwendung verfügt über einen PC mit dem Betriebssystem Windows. Die genutzten Versionen reichen von Windows 95 bis Windows 2000. Die Hardware reicht im Bereich der PCs von Intel 486er bis hin zu Ausstattungen aktueller Technologie. Ein kleiner Teil der Nutzer hat auf ihren Workstations aber auch Unix-Derivate installiert. In sehr wenigen Fällen hat eine bestellberechtigte Person keinen Computer zur Verfügung. In diesen Ausnahmefällen reagierte die Universität, indem sie die notwendigen Geräte und Netzwerkanbindungen zur Verfügung stellte.

Die Anbindung an das Universitätsnetz erfolgt in den meisten Fällen mit mindestens 100 MBit. In Einzelfällen verfügen ausgelagerte kleine Institute jedoch nur über eine ISDN-Einwahlverbindung.

Kenntnisse

Der Großteil der Anwender ist im Umgang mit den grundlegenden Windows-Funktionen und Office-Anwendungen erfahren. Ebenso können die meisten mit mindestens einem Browser und Email-Client umgehen und haben Erfahrungen in der Nutzung des Internets. Grundsätzlich reichen die Kenntnisse selten darüber hinaus. Die Softwareinstallation wird in der Regel von Mitarbeitern des Rechenzentrums durchgeführt. In Einzelfällen fehlen den potentiellen Nutzern einer Beschaffungsanwendung grundsätzliche Kenntnisse, nämlich dann, wenn sie keinen Zugang zu Computern hatten. Diese Probleme sollten sich mit der vorstehend beschriebenen Bereitschaft der Universität, Abhilfe bei diesen Infrastrukturproblemen zu schaffen, beseitigen lassen.

2.3 Die Lagerverwaltungssoftware HIS BEL

HIS BEL ist ein von der Firma *HIS Hochschul-Informations-System GmbH* ab 1990 entwickeltes Produkt zur Lagerverwaltung. Es wird vom Sachgebiet 12 seit 1991/92 zu diesem Zweck eingesetzt. Die HIS GmbH wurde 1969 zunächst von der Stiftung Volkswagenwerk gegründet und wird seit 1975/76 bzw. 1992 durch Bund und Länder finanziert, die auch die Gesellschafter der Firma sind. Die HIS GmbH wurde zum Zweck der Unterstützung der Hochschulen und der zuständigen Verwaltungen in ihrem Bemühen um eine rationale und wirtschaftliche Erfüllung der Hochschulaufgaben ins Leben gerufen [HI02].

2.3.1 Der technische Hintergrund von HIS BEL

HIS BEL wurde mit dem dBase-Compiler Clipper entwickelt. Nachdem das von der Firma Ashton Tate entwickelte Datenbanksystem dBase in den 80er Jahren die Marktführerschaft im Bereich der Personalcomputer gewann, wurde Clipper von der Firma Nantucket[9] entwickelt, um die Ausführungsgeschwindigkeit der für den dBase-Interpreter entwickelten Programme deutlich zu erhöhen.

Es konnte in dBase geschriebene Programme nach geringen Modifikationen kompilieren und nach dem Linken in eine ausführbare Datei umwandeln. Clipper entwickelte sich in den folgenden Jahren nach dem Aufkauf von Nantucket durch CA Computer Associates Anfang der 90er Jahre zu einem selbständigen Produkt, das 1996 in das Produkt VO Visual Objets von CA mündete. [Mü01]

HIS BEL läuft innerhalb einer Novell-DOS-Umgebung und bietet rudimentäre Multiuser-/Multisessionfähigkeiten. So haben mehrere Clients die Möglichkeit, HIS BEL auszuführen, wenn sie das auf dem Server freigegebene Verzeichnis der Software als Laufwerk verbunden haben. Die Sperrung von Datensätzen erfolgt durch den Client und wird in der Tabelle in einem Flag des jeweiligen Datensatzes abgelegt. Die Entscheidung, einen Datensatz zu sperren, wird durch die Laufzeitumgebung des Clientprogramms getroffen, aber nicht durch ein zentrales Datenbanksystem. Insofern handelt es sich bei HIS BEL aus Sichtweise der Schichten-Kategorisierung um eine *Ein-Schicht-Applikation*, die um rudimentäre Multiuser-Fähigkeiten erweitert wurde.

2.3.2 Erläuterung und Screenshots der benutzten Programmteile

Die Mitarbeiter der Beschaffung nutzen das Programm zur Verwaltung des Artikelstamms, Buchung von Artikelzugängen durch externe Lieferungen und Buchungen der universitätsinternen Bestellungen als Artikelabgänge an die jeweilige Kostenstelle der bestellenden Einrichtung. Daneben wird die Software genutzt, um etwa vierteljährlich eine Materialverbrauchsliste zu drucken, die den Einrichtungen zusammenfassend mitteilt, was sie in dem ausgewerteten Zeitraum bestellt haben.

[9] Einem im Internet kursierenden Gerücht zufolge (siehe dazu http://www.ghservices.com/gregh/clipper/story.htm) wurde Nantucket von zwei Mitarbeitern der Firma Ashton Tate gegründet. Sie hatten die Idee zu dem Compiler in dem von Ihnen häufig frequentierten Restaurant ‚Nantucket Lighthouse', daher nannten sie ihre Firma entsprechend. In dem Restaurant hing ein Bild eines Boots der Clipper-Klasse, windschnittig, elegant und schnell, das beide faszinierte. Diese Attribute assoziierten sie auch mit ihrer Idee und fanden damit den Namen für ihr Produkt.

Bild 1. Die Authentifizierung erfolgt in HIS BEL über eine zweistellige Sachbearbeiternummer und dem anschließend erfragten Namen

Bild 2 Es werden fast ausschließlich die unter dem Menüpunkt ‚Lagerwirtschaft' angebotenen Funktionalitäten genutzt

Bild 3. Im Bereich der Lagerwirtschaft werden hauptsächlich der Artikelstamm verwaltet, Materialbewegungen gebucht und die Listauswertungen (Materialverbrauchslisten) für die Einrichtungen erzeugt. Nach dem Aufruf des Menüpunkts ‚Artikelstamm' können Artikel nach Bezeichnung und Artikelnummer gesucht werden. Die Pflege ermöglicht die Neueingabe eines Artikels.

Bild 4. Nach der Auswahl des Artikels können alle Detailinformationen eingesehen werden. Änderungen in Preisen und Mengen werden nicht in diesem Dialog durchgeführt, sondern durch die Buchung von Zugängen und Abgängen. In dem Fall, dass die Artikelsuche mehrere passende Artikel findet, wird eine Liste der Treffer angezeigt und man gelangt in den dargestelltem Dialog durch die Auswahl einer der angebotenen Artikel.

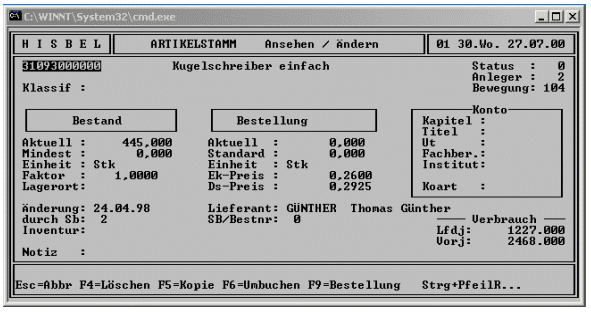


Bild 5. Zugänge und Abgänge von Artikeln werden unter dem Menüpunkt Materialbewegung zusammengefasst. Während es sich bei Zugängen um die Einbuchung von Artikeln handelt, die bei externen Lieferanten bestellt wurden, bezeichnet ‚Abgang' die Umbuchung von Artikeln aus dem Lager auf die Kostenstelle einer Einrichtung. Die Rückgabe ist die inverse Funktion zum Zugang, analog kann man mit der Stornierung falsch gebuchte Abgänge korrigieren.

Bild 6 Ein Materialzugang erfolgt immer an die Kostenstelle der Lagers. In diesem Fall werden 100 Kugelschreiber vom Lieferanten ‚ZIMO' in das Lager eingebucht. Der neue Preis wird als Durchschnittspreis aus altem und bestelltem Preis, gewichtet nach den Lager- und Bestellmengen ermittelt:

$$neu\Pr eis = \frac{(\Pr eis * Menge)_{Bestellung} + (\Pr eis * Menge)_{Lager}}{Menge_{Bestellung} + Menge_{Lager}}$$

Abbildung 2-2: Einige Screenschots der DOS-typischen Oberfläche von HIS BEL. Die Screenshots zeigen einige der hauptsächlich benutzten Funktionen der Software mit einer kurzen Erläuterung.

2.3.3 Nutzung im Sachgebiet 12 und Dokumentation

Wie der erste Screenshot andeutet, verfügt HIS BEL über eine einfache Nutzerverwaltung. Es gibt keine Möglichkeit; Nutzern verschiedene Rechte zuzuordnen. Jedoch können in HIS BEL mehrere Nutzer gleichzeitig arbeiten. Die in der Universität dazu aufgebaute Rechnerarchitektur besteht aus einem kleinen Netzwerk mit drei Computern. Der Server, auf dem sowohl die ausführbaren Dateien als auch die Datenbasis liegen, wird unter Novell betrieben. Jede der beiden Mitarbeiterinnen des Sachgebiets 12, die für die Lagerverwaltung und Materialausgabe zuständig sind, hat eine Workstation, die unter Windows 95 arbeitet. Sie haben ein Laufwerk F, das mit dem auf dem Novell-Server freigegebenen Verzeichnis verbunden wurde, unter dem die Anwendung im Verzeichnis /bel liegt. Der Aufruf einer Batch-Datei startet das Programm.

Die Dokumentation der Software ist spärlich und besteht im wesentlichen aus einer kurzen Beschreibung der im dBase-Format vorliegenden Tabellen. Sie erstreckt sich nicht auf ein Entity-Relationship-Modell, sodass sich die Beziehung der Tabellen untereinander nur unvollständig anhand des Namens ableiten lässt.

Aufgrund der schlechten Dokumentationslage wurde der Ansatz, die vorhandenen Datenstrukturen für die geplante internettaugliche Applikation zu verwenden, verworfen. Die Grundidee war, die weitere Nutzung des Legacy-Systems mit der erweiterten Funktionalität einer zusätzlichen Applikation zu verbinden.

Die Voraussetzungen scheinen auf den ersten Blick insofern gegeben zu sein, dass auf die verwendeten dBase-Tabellen direkt über einen *ODBC-Treiber*[10] zugegriffen werden kann. Aufgrund der fehlenden Informationen zu den Beziehungen unter den Tabellen konnte jedoch nicht garantiert werden, dass Datenänderungen über eine externe Applikation nicht zu Inkonsistenzen führen würden. Aus diesem Grund wurde der Ansatz zugunsten einer Neuentwicklung der Datenstrukturen nicht weiter verfolgt.

Über die *ODBC-Schnittstelle* wurde jedoch der Import der Artikelstammdaten in die neue Datenbankstruktur realisiert. Eine genauere Beschreibung hierzu folgt im Kapitel 3: Konzept und Implementierung.

2.4 Die Prozessketten in der Materialbeschaffung

Die Prozesse, die innerhalb der Einrichtungen, innerhalb des Sachgebiets 12 und zwischen den beiden Parteien ablaufen, sind über Jahre natürlich gewachsen. Es handelt sich weitgehend um universitätsinterne Prozesse. Sie berühren die Außenwelt an zwei Punkten.

[10] ODBC steht für Open Database Conectivity [JE93]. Das ODBC-Konzept wurde von der Firma Microsoft definiert. Es basiert auf dem SQL-CLI (Call Level Interface) der SQL Access Group.

Erstens sollten Materialbewegungen, die zu einer Unterschreitung der Mindestlagermenge führen, eine Nachbestellung auslösen. Wie dies zu erfolgen hat, wird durch Rahmenverträge definiert und soll entsprechend der Vorstellung des Sachgebiets 12 hier nicht weiter beschrieben werden (siehe Absatz 1.2.1).

Zweitens müssen nach erfolgreicher Bestellung Umbuchungen vom Konto der bestellenden Einrichtung auf das Konto des Sachgebiets 12 durchgeführt werden. Dies wird mit Hilfe eines Umbuchungsformulars realisiert, das im Original dem Sachgebiet 11 (Haushalt) zugeht, da dieser Vorgang in direkter Weise die vom Land Sachsen zugeteilten Mittel berührt. Umbuchungen dürfen ausschließlich mit Hilfe der Software MBS, die ebenfalls von der *HIS Hochschul-Informations-System GmbH* entwickelt wurde, durchgeführt werden. Dies ist dementsprechend die zweite Grenze, die eine neu zu erstellende Beschaffungsanwendung nicht überschreiten kann.

Innerhalb dieser Grenzen sollen in diesem Abschnitt die einzelnen Prozessketten dargestellt werden. Um den Arbeitsablauf möglichst genau wiedergeben zu können, werden ereignisorientierte und zyklische Prozesse unterschieden.

2.4.1 Ereignisorientierte Prozesse

Als ereignisorientierte Prozesse werden im Folgenden diejenigen Aktivitäten bezeichnet, die durch eine Bestellung ausgelöst werden und notwendig sind, um den Besteller in den Besitz des gewünschten Materials zu bringen.

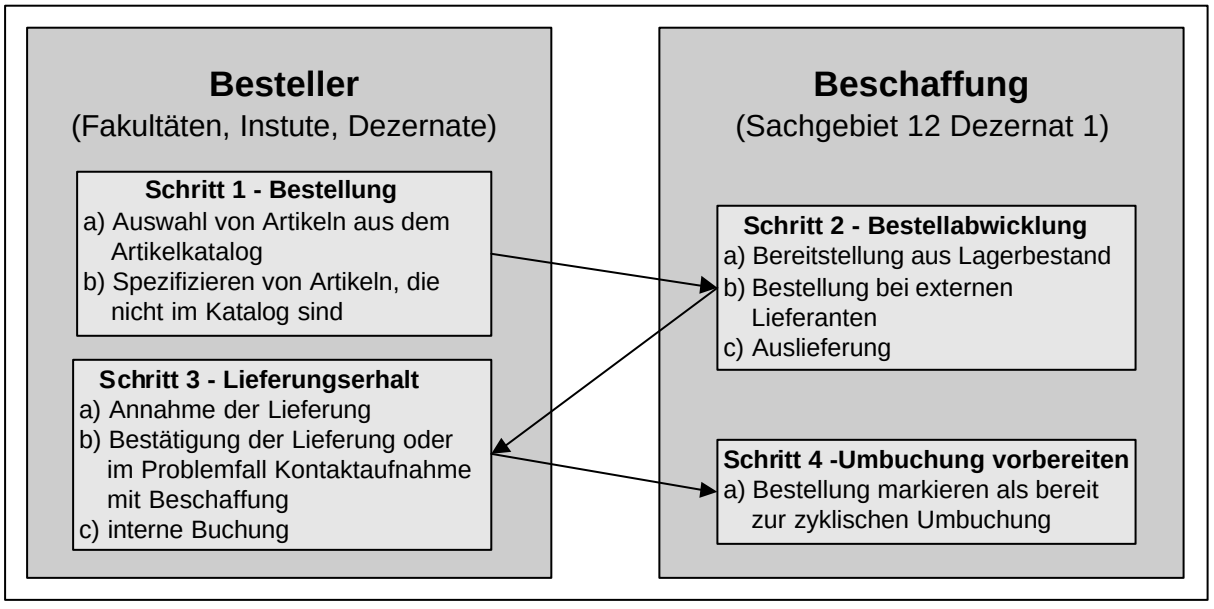


Abbildung 2-3: Vier Schritte bilden den ereignisorientierten Teil der Prozesskette einer Bestellung

Unter normalen Bedingungen wird eine solche Prozesskette innerhalb von zwei bis drei Tagen abgearbeitet.

2.4.1.1 Schritt 1: Bestellung:

Handelt es sich um klassische C-Güter, so findet der Bestellende in den meisten Fällen das von ihm gesuchte oder ein vergleichbares Produkt im Katalog. In diesem Fall müssen Artikelnummer, Menge und Bezeichnung sowie eine optionale Bemerkung in das von der Beschaffung bereitgestellte Formular (siehe Abbildung 2-1) übertragen werden. Die Artikelnummer ist neunstellig.

Die ersten drei Stellen geben die Kostenart des Artikels an. Die folgenden sechs Stellen werden von der Beschaffung frei vergeben. Eine eindeutige Regel ist dabei nicht zu erkennen. In den meisten Fällen werden die ersten drei Stellen des Sechsstellers als eine Art interne Untergruppe der Kostenart gewählt, die folgenden beiden Stellen als fortlaufender Zähler und die letzte Stelle wird für den Fall reserviert, dass eine Artikelnummer zwischen zwei andere direkt aufeinander folgende eingefügt werden soll. Falls der fortlaufende Zähler (Stelle 4/5 des Sechstellers) nicht ausreicht, gibt es Fälle, in denen der Überlauf in die interne Untergruppe übertragen wird.
Bei selten angefragten und häufiger bei B-Artikeln trägt er in wie bei den aus dem Katalog gewählten Artikeln Stückzahl und Bezeichnung ein, ergänzt aber in den meisten Fällen zusätzlich die Information eines Preises, den er bereits bei einem kommerziellen Verkäufer erfragt hat.

2.4.1.2 Schritt 2: Bestellabwicklung

Nach Eingang der Bestellung wird im Artikelstamm von HIS BEL nachgeprüft, ob der bestellte Artikel bekannt ist. Falls dies der Fall ist, wird geprüft, ob er in der erforderlichen Stückzahl bereitgestellt werden kann. Ist der Artikel nicht bekannt oder die Lagermenge nicht ausreichend, wird der Artikel in den meisten Fällen über einen Lieferanten bestellt. Falls noch eine Restmenge im Lager vorhanden ist, wird meist die Bestellmenge auf der Bestellung entsprechend korrigiert, die Änderung kenntlich gemacht und um eine Nachbestellung zu einem späteren Zeitpunkt gebeten. Falls der Artikel nicht mehr lieferbar ist, oder nicht zu den Artikelgruppen gehört, deren Beschaffung Aufgabe des zentralen Lagers ist (bspw. Geräte und Ausstattungen), wird ein entsprechender Vermerk auf der Bestellung gemacht.

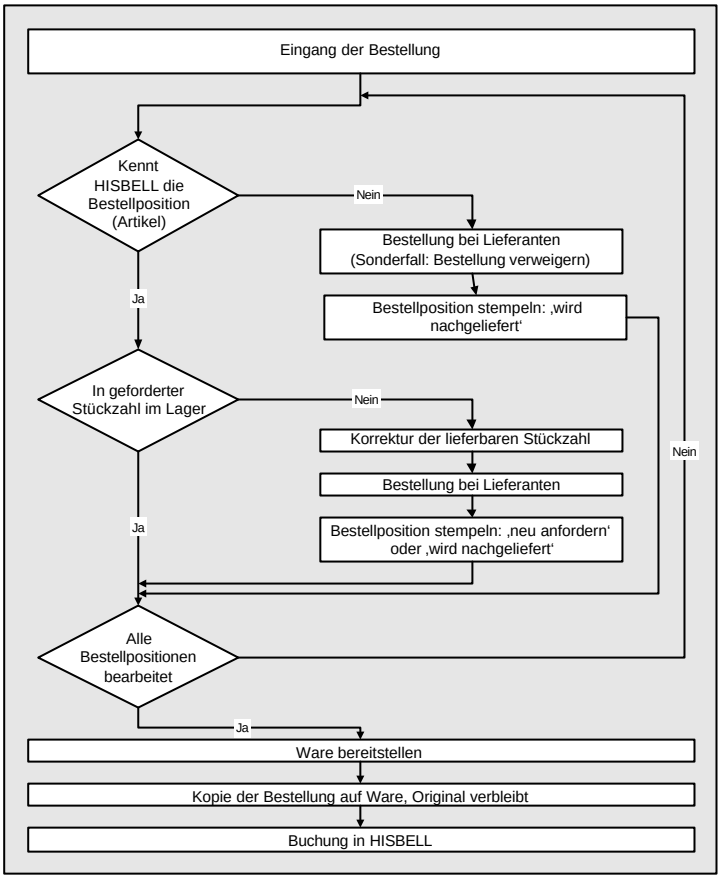


Abbildung 2-4: Beschreibung der Arbeitsschritte in der Beschaffung, die durch den Erhalt einer Bestellung initiiert werden

Eine nachbestellte Position wird auf dem Bestellformular mit dem Stempel: ‚wird nachgeliefert' versehen. Sobald der Artikel von einem der Lieferanten geliefert wurde, erfolgt automatisch die Weiterleitung an die Einrichtung. In der Praxis gestaltet sich dieser Vorgang oftmals als ein Reibungspunkt zwischen des Bestellern und der Beschaffung. Die Anzahl der nachzuliefernden Artikel ist aufgrund der Begrenzungen des Lagers vergleichsweise hoch. So liegen zu einem beliebigen Zeitpunkt im Jahr zwischen 50 und 200 Nachbestellungen vor.

Da es kein technisches System gibt, dass diese Nachbestellungen mit den Artikeln der Lieferungen vergleicht, um nachzuliefernde Positionen zu identifizieren, muss dieser Vorgang manuell durchgeführt werden. Da die Mitarbeiter der Beschaffung im Rahmen des Tagesgeschäfts kaum die notwendige Zeit finden, verzögern sich die Auslieferungen um einen längeren Zeitraum,

Eine ähnliche Situation ergibt sich bei der Abarbeitung der nächsten Sequenz der Abbildung. Falls der Bestand des Lagers nur eine Teillieferung einer Bestellposition ermöglicht, wird ebenfalls eine Nachlieferung nötig. Um den Aufwand zu verringern, entscheiden sich die Mitarbeiter hier oftmals für die Variante: ‚neu anfordern', die den Besteller auffordert, den Artikel noch einmal zu bestellen. Diese Möglichkeit wird in den meisten Fällen dann gewählt, wenn die Mitarbeiter des Sachgebiets 12 die der Einrichtung gelieferte Teilmenge im Rahmen des aktuellen Bedarfs für ausreichend halten.

Selbstverständlich ist dies Gegenstand einer subjektiven bzw. willkürlichen Entscheidung und damit auch Gegenstand von Auseinandersetzungen zwischen den Parteien.
Die Lagerabgänge werden in HIS BEL auf die Kostenstelle des Bestellers gebucht, physisch zusammengestellt und von einem Mitarbeiter zum Besteller gebracht.
In Einzelfällen benötigen Einrichtungen bestellte Materialien umgehend. Dies führt in einem Teil der Fälle dazu, dass Artikel ausgegeben werden, ohne dass sie bereits in HIS BEL gebucht wurden. Die Lagerbewegung wird erst in den Folgetagen nachgeholt.

2.4.1.3 Schritt 3: Lieferungserhalt

Der Besteller kontrolliert die Waren in Qualität und Anzahl auf Korrektheit. Falls es Abweichungen gibt, kontaktiert er das Sachgebiet 12, um gemeinsam nach einer Erklärung zu suchen. Eventuelle Abweichungen von der Bestellung werden vom Sachgebiet 12 vermerkt, um zu garantieren, dass in die zyklisch erfolgende Abrechnung mit den Einrichtungen keine falschen Werte eingehen.
Falls Bestellpositionen mit ‚neu anfordern' markiert wurden, löst er die Neubestellung der betreffenden Artikel mit der nächsten Bestellung wieder aus.
Einige Einrichtung führen auf einer zentralen Kostenstelle Sammelbestellungen für die ihr untergeordneten Einrichtungen durch. Ist diese Situation gegeben, muss die übergeordnete Einrichtung eine Umbuchung auf die ursprünglichen Besteller und Empfänger der Waren durchführen. Dazu setzen sie eine in der Funktionsvielfalt reduzierte Version des Programms MBS, das sogenannte HIS MBS – PC ein.
Der korrekte Wareneingang wird vom Besteller durch Unterschrift auf dem beiliegenden Lieferschein als Kopie der Bestellung bestätigt. Die Bestätigung wird vom Auslieferer zurück zum Lager gebracht.

2.4.1.4 Schritt 4: Vorbereitung der Umbuchung

Die Mitarbeiter des Lagers im Sachgebiet 12 nehmen die Bestätigung der korrekten Lieferung entgegen und legen sie ab. Diese gesammelten Belege sind die Basis für den alle zwei bis drei Monate erfolgenden zyklischen Prozess der Umbuchung (siehe 2.4.2.4). Falls die Mitarbeiter durch die vorstehend beschriebene Situation keine Bestätigung des Wareneingangs bekommen, gilt der Grundsatz, dass die Ware ordnungsgemäß ausgeliefert wurde. Eine eventuell auftretende Abweichung der Lieferpositionen von den bestellten Artikeln und Reklamationen anderer Art werden auf direktem Wege telefonisch bereinigt.

2.4.2 Zyklische Prozesse

Das Sachgebiet 12 kennt verschiedene zyklische Prozesse. Sie unterscheiden sich durch die unterschiedlichen zeitlichen Abständen, in denen sie ausgelöst werden. Dazu gehören beispielsweise die Aufbereitung des Artikelkatalogs alle zwei Jahre in Folge der im gleichen Rhythmus erfolgenden

Ausschreibungen oder die etwa alle zwei bis drei Monate durchgeführte Erstellung der Materialverbrauchslisten als Information für die Einrichtungen.

2.4.2.1 Aktualisierung des Artikelkatalogs

Die Aufbereitung des Artikelkatalogs als Wordperfect-Datei wird vom Sachgebiet 12 etwa alle zwei Jahre durchgeführt, um anschließend an alle Fakultäten, Institute und Dezernate verschickt zu werden. Er beinhaltet in 11 Gruppen unterteilt alle etwa 1000 bestellbaren Artikel mit Informationen zur Bezeichnung des Artikels und zum Preis.

Der Zyklus ergibt sich aus den im gleichen Abstand durchgeführten Ausschreibungen für die in Absatz 2.1 genannten Artikelgruppen.

Nach erfolgter Entscheidung für die Lieferanten der Artikelgruppen muss der Artikelkatalog angepasst werden, um die Einrichtungen über die neuen Artikel und Preise zu informieren. Da der Aufwand einer häufigeren Anpassung des Artikelkatalogs zumindest bezüglich der Artikelpreise und seiner anschließenden Versendung an alle Einrichtungen groß ist, haben sich alle Beteiligten auf den Konsens geeinigt, Abweichungen zwischen dem Bestellwert und dem zyklisch abgerechneten Wert innerhalb bestimmter Grenzen zu tolerieren.

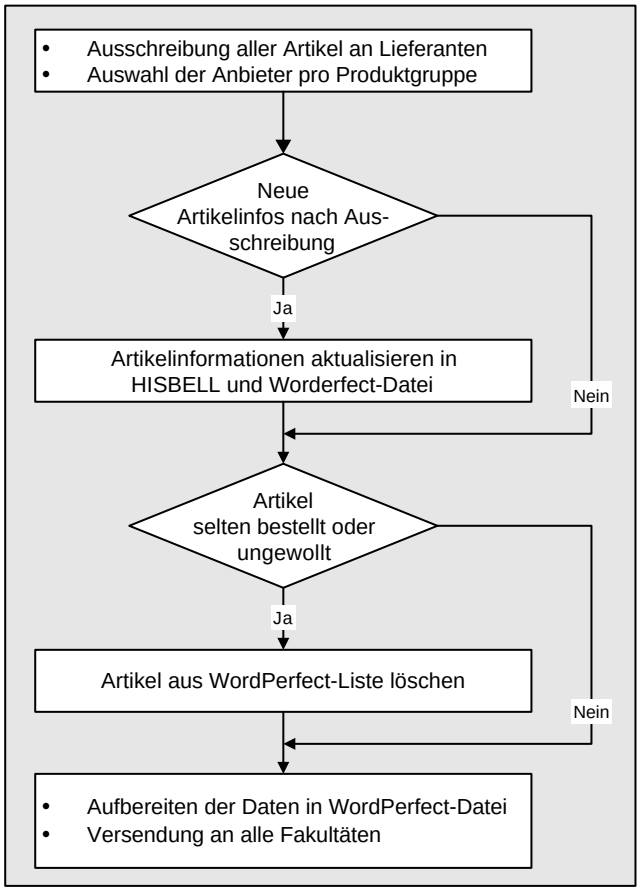


Abbildung 2-5: Flußdiagramm der in zweijährigem Rhyhtmus durchgeführten Aktualisierung des Artikelkatalogs

Ein Hauptnutzen des Artikelkatalogs liegt aus Sicht des Sachgebiets 12 in seiner normierenden Wirkung. Die Beschaffung sucht aus den Katalogen der Lieferanten diejenigen Artikel aus, welche in einem guten Preis-/Leistungsverhältnis stehen. Für diese wird aufgrund der hohen Abnahmemengen ein deutliche Rabattierung von bis zu 33% verhandelt. Indem die Einrichtungen diese Artikel bestellen, erhalten sie nicht nur die niedrigen Artikelpreise, sondern orientieren sich auch an der vorgeschlagenen Artikelauswahl, kaufen also keine extravaganten oder überteuerten Artikel.
In Einzelfällen setzen sich Einrichtungen über die Empfehlungen der Dezernats 1, Material über das zentrale Lager zu bestellen, hinweg und führen ihre Materialbeschaffung selbst durch.

2.4.2.2 Kontrolle und Bearbeitung von Nachlieferungen

Nachlieferungen werden nötig, wenn eine Bestellposition wie in Absatz 2.4.1.2 beschrieben nicht oder nur in einer Teilmenge bedient werden kann, weil der Lagerbestand nicht ausreicht.
Die Bestellungen, in denen mindestens eine Bestellposition aus den genannten Gründen nicht ohne Verzögerung bearbeitet werden kann, werden gesammelt. Es wird kein System eingesetzt, das es ermöglicht, die Positionen von eingehenden Lieferungen externer Lieferanten mit den offenen Bestellpositionen zu vergleichen. Deshalb werden im Rhythmus von zwei bis sechs Wochen alle noch offenen Bestellpositionen gegen den durch Lieferungen ergänzten Bestand in HIS BEL geprüft. Falls der Bestand die Lieferung ermöglicht, wird diese abschließend bearbeitet und die Ware ausgeliefert. Falls nicht, wandert die Bestellung wieder zurück auf den Stapel nachzuliefernder Bestellungen. Der Mitarbeiter der Beschaffung sollte pro betroffener Bestellposition nachprüfen, ob die vom Lieferanten angegebene Lieferzeit überschritten ist. Aufgrund der Vielzahl der Nachlieferungen findet er dazu nur in Einzelfällen die Zeit.
Wird der Sachbearbeiter vom Lieferanten kontaktiert, weil eine Bestellung durch den Lieferanten nicht durchgeführt werden kann, bspw. weil der Artikel nicht mehr im Sortiment ist, kann der Sachbearbeiter entweder einen anderen Lieferanten mit der Bereitstellung beauftragen oder den Vorgang beenden, indem er der Einrichtung mitteilt, dass der Artikel nicht mehr beschafft werden kann.

2.4.2.3 Erstellen und Versenden der Gesamtverbrauchsliste

Im Abstand von zwei bis drei Monaten erstellt das Sachgebiet 12 unter Einsatz des Programms HIS BEL eine sogenannte Gesamt- oder Materialverbrauchsliste. Sie listet alle Materialbewegungen, die in HIS BEL eingegeben wurde gruppiert nach den Kostenstellen, an die Material geliefert wurden, auf. Zu jeder Kostenstelle wird zusätzlich eine Summe gebildet. Diese Liste wird den Einrichtungen zur Kontrolle übersendet und bildet die Basis für die anschließende Umbuchung der von der Beschaffung für die Lagerbereitstellung vorfinanzierten Artikel auf die Kostenstellen der Einrichtungen. Eine Unterteilung der in der HIS BEL Materialverbrauchsliste gelisteten Artikel in Kostenarten ist nicht möglich.

2.4.2.4 Umbuchung zwischen Verwaltungshaushalt und Einrichtungen

Im Nachgang der Erstellung der Gesamtverbrauchsliste und unter der Bedingung ihrer Akzeptanz durch die Einrichtung nehmen die Mitarbeiter des Sachgebiets 12 eine kostenstellenweise Umbuchung vor. Die Wirkung der Umbuchung ist eine Entlastung der Kostenstelle des Lagers, welches die Beschaffung der Artikel vorfinanziert hat, und eine Belastung der Kostenstelle der bestellenden Einrichtung. Weil die Umbuchung in sich eine eigene Prozesskette mit mehreren Schritten darstellt, wird sie nachfolgend schematisiert:

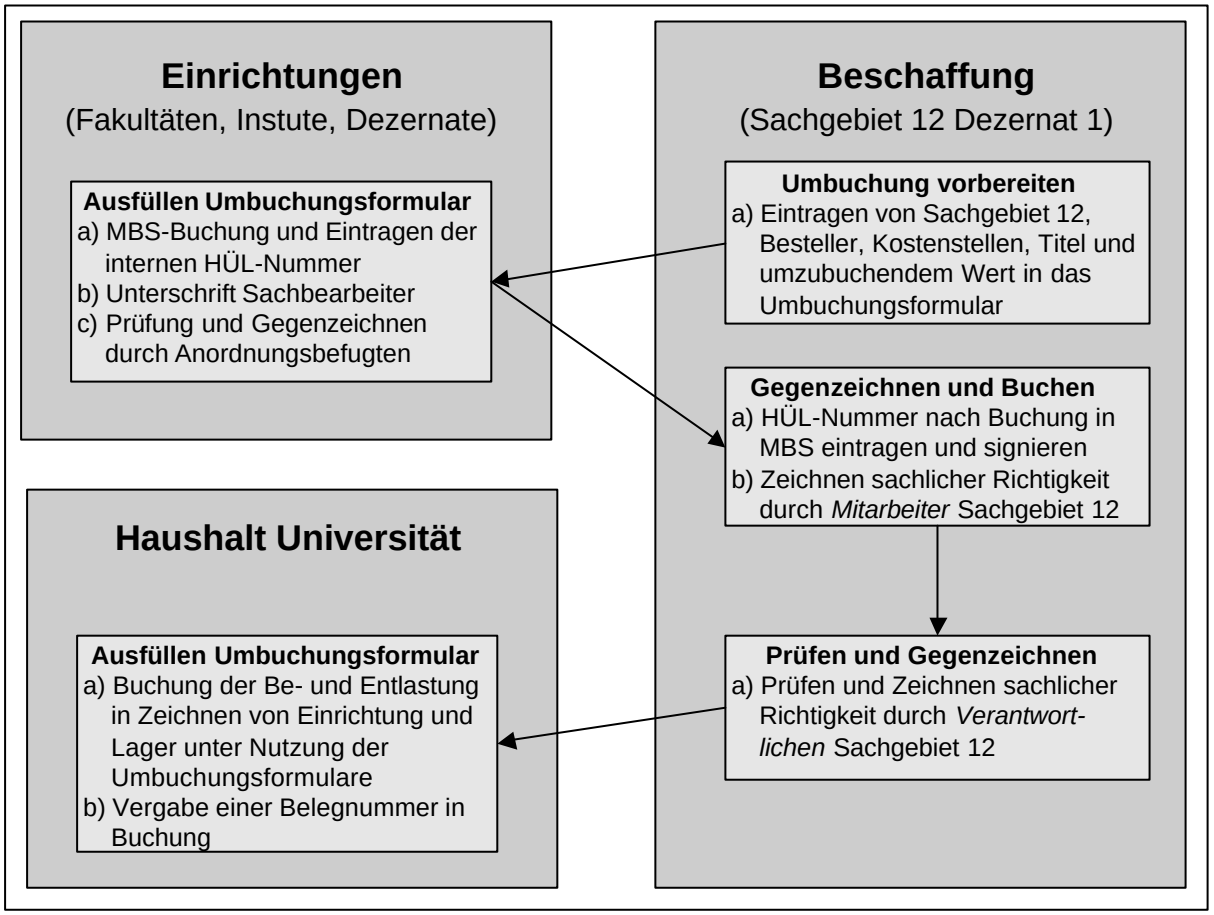


Abbildung 2-6: Prozesskette der Umbuchungen. Die HÜL-Nummer ist eine vom MBS-Programm pro Buchung vergebene Identifikationsnummer.

Die gesamte dargestellte Prozesskette wird über ein Umbuchungsformular abgewickelt, das beispielhaft unter Anhang D zu finden ist. Auf diesem Formular werden bis zur vollständigen Bearbeitung einer Bestellungsumbuchung sechs Unterschriften von insgesamt 4 Personen und 3 Dienststellen geleistet.

2.5 Ansatzpunkte für die Prozessoptimierung

Betrachtet man die beschriebenen ereignisorientierten und zyklischen Prozesse aus dem Blickwinkel der konzeptionellen Planung einer unterstützenden Applikation und untersucht sie auf vorhandene Schwachstellen, ergibt sich eine Vielzahl von Ansatzpunkten für Verbesserungen.
Nachstehend werden diese Ansatzpunkte inhaltlich geordnet aufgeführt und jeweils mit einer abgeleiteten Forderung für das Konzept einer neuen Applikation ergänzt.

Aufbau klarer Datenstrukturen

A. Die Trennung von Bestellungen des Bewirtschaftungsbedarfs und Bürobedarfs (Absatz 2.1) ist aus Sicht der Einrichtungen doppelter Aufwand und nur schwer verständlich zu machen. Grundsätzlich sollte ein zukünftiges System die Trennung aufheben.
B. Die Nutzerverwaltung von HIS BEL bietet keine Möglichkeit an, die von Bestellern zu manipulierenden Datenbestände einzuschränken. Unter dem Gesichtspunkt eines Direktzugriffs auf Bestelldaten sowohl durch verschiedene Nutzergruppen innerhalb der Einrichtungen als auch durch mit abweichenden Rechten versehene Nutzer des Sachgebiets 12 sollte das zu entwickelnde Zugriffssystem in Abhängigkeit der einem Nutzer zugeordneten Rolle gesteuert werden und das Auslösen/Verwalten von Bestellungen sowie das Generieren von Auswertungen nur für ihm zugeordnete Kostenstellen durchführen können.
C. Die in Absatz 2.4.1.1 beschriebene nicht einheitliche Vergabe von Artikelnummern sollte zugunsten eines klaren und auch zukünftig nicht beschränkenden Algorithmus modifiziert werden.

Transparenz der Prozesse

D. Ist die Lagerbestandsmenge eines Artikeln so gering, dass nicht alle Bestellungen vollständig bedient werden können, wird die Entscheidung, welchen Bestellungen wie viel Stück des knappen Artikels zugeteilt werden, willkürlich getroffen. Das hat in Einzelfällen den Vorwurf der Begünstigung bestimmter Einrichtungen nahegelegt. Der Weg zu einer Verbesserung dieser Konfliktsituation soll in der Applikation durch eine abschaltbare Einsicht in die Bestandmengen der Artikel begegnet werden. Die Einrichtungen erhalten ausschließlich eine Information darüber, ob ein Artikel verfügbar ist, aber nicht die genaue Höhe des Artikelbestands. Die fehlende Detailauskunft soll hier nach Überzeugung des Sachgebiets 12 zu einer Reduktion des Konfliktpotentials führen.
E. Die in Absatz 2.4.1.2 beschriebenen Wartezeiten bei Nachbestellungen sind mit bis zu 3 Monaten zu hoch. Eine zu entwickelnde Anwendung muss die Mitarbeiter des Sachgebiets 12 in diesem Punkt so unterstützen, dass die Wartezeiten drastisch reduziert werden.

F. Die Möglichkeit einer zeitlich nachgelagerten Buchung von Materialbewegungen in HIS BEL (Absatz 2.4.1.2) erhöht das Risiko von Fehlern in der Bearbeitung von Bestellungen. Die Prüfung auf ausreichende Bestandsmenge zur Erfüllung einer Bestellung, ihre Buchung und die Einsicht in parallel existierende Bestellungen sollen so stark verzahnt sein, dass die asynchrone Abarbeitung einer Bestellung als Mehraufwand verstanden und deshalb unsinnig wird.

G. Die Bestätigung der korrekten Lieferung einer Bestellung wird durch die belieferte Einrichtung vorgenommen. In der Praxis tritt sporadisch der Fall auf, dass der Auslieferer niemanden antrifft, der ihm die Bestätigung gegenzeichnen kann. Trotzdem verbleibt die Ware beim Besteller. Ein zu entwickelndes System sollte eine klare Regelung für diesen Fall abbilden können.

H. Die Behandlung von Reklamationen und eventuell daraus folgenden Stornierungen wird zum größten Teil direkt zwischen den Beteiligten geklärt, ohne dass eine Protokollierung erfolgt. Eine zu entwickelnde Applikation sollte hier ein nachvollziehbares Vorgehen ermöglichen.

Übersichtlichkeit der Anwendung

I. Das Problem der schlechten Lesbarkeit von handgeschriebenen Bestellungen soll durch die Nutzung von Eingabemasken eliminiert werden

J. Ein neues System in der Lage sein, Nachbestellungen so zu präsentieren, dass alle Bestellungen, die einen gleichen nachzuliefernden Artikel beinhalten, eingesehen werden können. Dies bewirkt eine weitere Beschleunigung in der Abarbeitung von Nachbestellungen.

K. In 2.4.2.4 wird deutlich, dass ein vollständiger Beschaffungsprozess inklusive der Umbuchung eines Artikels die Aktivität einer Vielzahl von Personen erfordert. Es ist Aufgabe eines neuen Systems zu prüfen, an welcher Stelle diese Anzahl reduziert werden kann, bspw. durch die Optimierung des Umbuchungsprozesses.

Echtzeitzugriff auf Daten und bereitgestellte Auswertungen

L. Die Aktualisierung des Artikelkatalogs im Rhythmus von etwa 2 Jahren ist zu lang. Die lange Wartefrist führt allein aufgrund der allgemeinen Preissteigerung schon zu falschen Preisen. Preisschwankungen können den Einrichtungen erst zum Zeitpunkt der Buchung bekannt gegeben werden und führen bei diesen zu teilweise gravierenden Problemen bei der Budgetplanung, da die wirklichen Kosten von den geplanten stark abweichen. Des können die Einrichtungen nicht über Änderungen im Sortiment unterrichtet werden, es sei denn durch die persönliche Unterrichtung durch Nachfragen, die den Arbeitsaufwand der im Sachgebiet 12

wiederum erhöht. Eine zu entwickelnde Applikation soll deshalb einen in Echtzeit verwalteten und einsehbaren Artikelkatalog beinhalten.

M. In Vorbereitung der Umbuchungen der Beschaffung auf die Kostenstellen der Einrichtungen ist die von HIS BEL erzeugte Materialverbrauchsliste ungenügend. Sie soll durch die Fakultäten zu jedem Zeitpunkt selbst druckbar sein. Unabdingbar ist die Unterteilung der gelieferten Artikel in Kostenarten, da ein zu entwickelndes System in der Lage sein muss, die von der Kosten- und Leistungsrechnung geforderte Umbuchung nicht per Kostenstelle, sondern per Kostenart einer jeden Kostenstelle, abzubilden. Ebenso ist der zeitliche Abstand zwischen erfolgter Lieferung und Zustellung der Gesamtverbrauchsliste so groß, dass die Ursachenforschung einer möglichen Abweichung erschwert wird.

Die vorstehenden Punkte sind direkt aus den beschriebenen Prozessen abgeleitete Anforderungen an die zu entwickelnde Applikation. Sie werden ergänzt durch die im Absatz 1.2.2 genannten Anforderungen.

Im folgenden Kapitel werden das aus der Summe an Forderungen und den analysierten Prozessen entwickelte Konzept und seine Implementierung vorgestellt.

3 Konzept und Implementierung

Die durchgeführten Analysen ergeben eine fundierte Basis für die Konzeption der entwickelten Anwendung. Eine zentrale Frage ist jedoch noch vor der Beschreibung der konzeptioneller Eckpunkte zu beantworten: Auf Basis welcher Architektur lässt das Konzept der Anwendung am erfolgreichsten entwickeln ? Die Beantwortung dieser Frage wurde bereits im Rahmen der Positionierung der vorliegenden Arbeit in der Einleitung vorweggenommen. Die folgenden Absätze liefern die Begründung für diese Entscheidung. Ausgehend von den Eigenschaften der gewählten Architektur werden die Berührungspunkte mit den geforderten Merkmalen der Beschaffungsanwendung aufgezeigt.

3.1 Wahl der Architektur und Entwicklungsumgebung

Die im Rahmen der vorliegenden Arbeit entwickelte Applikation implementiert eine Drei-Schicht-Architektur. Mit ihrer Hilfe lässt sich aus Sicht des Autors erfolgreich das Anforderungsprofil umsetzen, welches aus den im ersten Kapitel (Absatz 1.2.2) genannten Zielstellungen und den Ergebnissen der Analyse resultiert.

3.1.1 Drei-Schicht-Architektur

Die vorstehend genannte Drei-Schicht-Architektur ist geeignet zur Realisierung verteilter und skalierbarer Anwendungen. Die Abbildung 3-1 skizziert das Zusammenspiel von Client-Schicht und Datenbanksystem mit der mittleren Schicht, in der sich neben dem Webserver

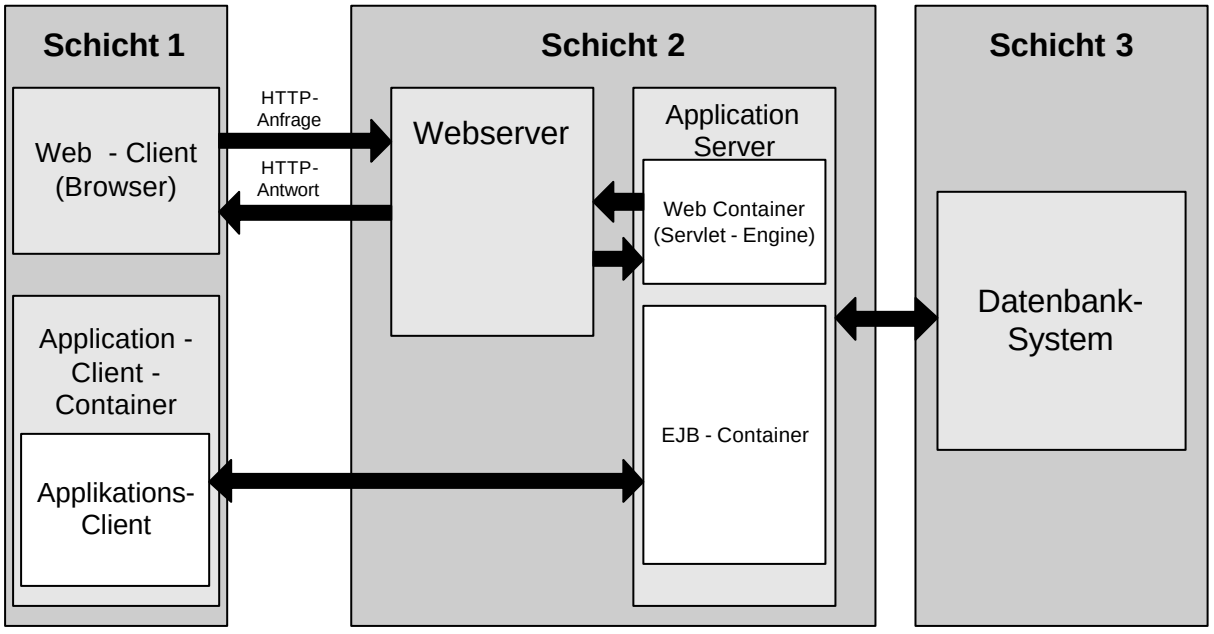


Abbildung 3-1: Drei-Schicht-Architektur mit Client, Applikationslogik und Datenbanksystem

insbesondere der die Programmlogik ausführende Application-Server positionieren. In der Schicht 1 dient der Browser als Anwender-Interface. Der Nutzer generiert bspw. durch das Klicken auf einen Hyperlink oder Submit-Button eine Anfrage. Sie wird vom Browser per HTTP-Protokoll an den Webserver geschickt. Der Webserver prüft, ob es sich bei der Anfrage um eine statische oder dynamische Seite handelt. Statische Seiten, also schon im HTML-Format vorliegende Internetseiten kann er sofort an den Browser zurückschicken. Requests für dynamische, also zu erzeugende Seiten, werden an den Application-Server weitergereicht. Er erzeugt möglicherweise unter Nutzung des Datenbanksystems die HTML-Seite und schickt sie an den Webserver zurück, der sie seinerseits an den Browser ausliefert. Eine weitere Spielart der Drei-Schicht-Architektur im Rahmen der J2EE-Spezifikationen ist die alternative Entwicklung eines Application-Clients, der direkt mit dem EJB-Container des Application-Servers kommuniziert. Dieser Ansatz wird in der vorliegenden Arbeit nicht verfolgt, da er zu der Entwicklung eines auf dem Rechner des Nutzers zu installierenden Anwendung führen würde. Die Gründe der Entscheidung für die Nutzung eines Browsers als Nutzerinterface und die vorstehend skizzierte Drei-Schicht-Architektur werden in den folgenden Absätzen dargelegt.

Die Argumentation für die Drei-Schicht-Architektur wird im folgenden beginnend mit dem Nutzerfrontend (Schicht 1) aufgebaut. Als Voraussetzung für diese Diskussion kann man die Notwendigkeit einer zentralen Datenhaltung (Schicht 3) als gegeben betrachten, weil eine große Anzahl von Nutzern in Echtzeit in lesendem und schreibenden Zugriff auf gleiche Daten zugreifen will. Das kann nur durch ein zentrales Datenbanksystem geleistet werden.

3.1.1.1 Thin-Client statt Fat-Client

Grundsätzlich kann eine Anwendung auf dem Computer des Nutzers installiert sein und sich mit einer zentralen Datenquelle bzw. einem Application-Server verbinden oder der Nutzer steuert gleichsam ein Programm fern, das aber ausschließlich auf einem entfernten Server ausgeführt wird. Der zuerst geschilderten Fall wird als *Fat- oder Rich-Client* bezeichnet, letzterer als *Thin-Client.*
Die Bestandsaufnahme und Forderungen des Sachgebiets 12 lassen sich wie folgt zusammenfassen:

- Der Wartungsaufwand des durch den Nutzer eingesetzten Clients soll möglichst gering sein.
- Die bei den meisten Nutzern vorhandenen Kenntnisse reichen nicht für die Installation von neuer Software aus.
- Das Rechenzentrum hat nicht ausreichend Personal, um auf jedem der 250 PCs (siehe Abschnitt 2.2) die Neuinstallation eines Programms durchzuführen.
- Die einfache Handhabbarkeit und damit die Akzeptanz des Systems soll gewährleistet sein, indem sich das Programm an bekannten Anwendungen orientiert.
- Das Nutzerinterface muss auch auf langsameren Prozessoren ergonomische Antwortzeiten haben.

Die Summe dieser Forderungen zeigt deutlich die Notwendigkeit des Einsatzes eines browserbasierten Thin-Clients an. Er enthebt den Nutzer von der Notwendigkeit einer Installation einer weiteren Software, muss nicht vom Nutzer oder einer externen Instanz gewartet werden und läuft unter allen Betriebssystemen. Somit spart er Geld in der Wartung und in der Entwicklung. Weiterhin sollte auf die Nutzung von Java-Applets verzichtet werden, da diese in ihrem Laufzeitverhalten stark abhängig von der Rechenleistung des dem Nutzer zur Verfügung stehenden Rechners und der auf dem Rechner installierten Java-Version sind. Man kann diese Nachteile und Unwägbarkeiten ausklammern, indem man sich als Thin-Client der Browser mit der Seitenbeschreibungssprache HTML (HyperText Markup Language) und der für sie definierten Standards bedient. Hinzu kommt, dass alle Nutzer, die bereits Kontakt mit Internetseiten hatten, schnell mit dem System zurecht kommen können, da ihnen die Navigation im Internet geläufig ist.
Aus der Entscheidung für einen HTML-basierten Thin-Client folgt, dass *die Intelligenz der Anwendung auf der Seite des Servers*, der die HTML-Seiten an den jeweiligen Client ausliefert, implementiert werden muss, weil HTML als Seitenbeschreibungssprache keine Algorithmen ausführen kann, sondern nur ihre Ergebnisse zur Anzeige bringt.

3.1.1.2 Javabasierte Application-Server versus CGI, PHP/ASP und .NET

Aus der im vorstehenden Absatz festgestellten Notwendigkeit des Einsatzes eines browserbasierten Thin-Clients folgt, dass der Programmcode der Anwendung auf einem zentralen Server ausgeführt werden muss. Für diesen Anwendungsfall bieten sich verschiedene Varianten an. Eine der ältesten Ansätze ist die Nutzung des von der NCSA (National Center for Supercomputing Applications) definierten *CGI (Common Graphic Interface)* in Kombination mit beliebigen Programmiersprachen. Die einzige Einschränkung ist, dass die in der jeweiligen Sprache geschriebenen Programme im Batch ausführbar sind, und man ihnen Parameter mitgeben kann. Der Ansatz des CGI ist es, einem Nutzer durch seinen Browser die Möglichkeit zu geben, ein auf dem zentralen Server befindliches Programm mit Parametern zu starten und die Ausgabe des Programms wieder im Browser anzuzeigen. Der Webserver empfängt die Anforderung eines Nutzers, und führt das Programm mit den vom Nutzer übergebenen Parametern aus. Das Programm generiert daraufhin Informationen, die vom Browser angezeigt werden können und schickt sie an den Webserver, der sie an den Nutzer zurückreicht. Die Nachteile dieses Vorgehens ergeben sich aus verschiedenen Gründen. Für jeden einzelnen Request wird ein CGI-Programm ausgeführt. Das bedeutet, es entsteht für jeden Request Overhead. Ein Prozess muss erzeugt werden, und ihm müssen Ressourcen des lokalen Rechners, auf dem er ausgeführt wird, zugeordnet werden. Das Programm selbst muss bei jeder Ausführung eine Datenbankverbindung herstellen und verwalten. Die Kosten für diese Prozesse sind sowohl für den Server als auch für das Datenbanksystem, das unter Umständen auf dem gleichen Server läuft, hoch.

Einen anderen Ansatz verfolgten Microsoft mit *ASP (Active Server Pages)*, Netscape mit *SSJS (Server-Side JavaScript)* und die open-source-Gemeinde mit *PHP (Hypertext Preprocessor, ehemals Personal Homepage Tool)*. Gemeinsam ist diesen Ansätzen, dass vom Webserver über ein Application Programming Interface eine Runtime Engine angesprochen wird, die über das CGI-Konstrukt hinausgehende Dienste anbietet. Die Runtime Engines der verschiedenen Hersteller bieten den in ihnen laufenden Anwendungen beispielsweise Dienste wie die Unterstützung bei der Herstellung von Datenbankverbindungen an, E-mail-Support oder Session-Management. Die jeweiligen Dienste sind optimiert in Ausführungsgeschwindigkeit und Sicherheit. Sie vergrößern die Ausführungsgeschwindigkeit der Anwendungen und verkürzen die Anwendungsentwicklung. Des weiteren ist der Overhead beim Ausführen von Anwendungen in der Runtime Engine geringer als beim CGI-Modell. Die von einem Thread angeforderten Ressourcen werden von der Runtime Engine bereits beim Systemstart alloziiert und müssen nur noch an den Thread übergeben werden, also nicht vom Betriebs- oder Datenbanksystem neu angefordert werden.

Der Nachteil dieser Lösung liegt in der engen Verbindung von HTML-Code und Programmierung. Die auszuführenden Programme enthalten gemischt sowohl HTML-Code wie auch Codefragmente der von der jeweiligen Runtime Engine akzeptierten Programmiersprache, im Fall von Active Server Pages beispielsweise Visual Basic. Nachteilig ist dies für die Entwicklung einer Webanwendung insofern als dass es *keine klare Trennung zwischen Anwendungslogik und Design* einer Internetseite gibt. Dies macht sowohl die Entwicklung als auch die Wartung der jeweiligen Seiten aufwendig, denn der Programmierer muss nicht nur die Anwendungslogik implementieren, sondern auch gleichzeitig immer die HTML-spezifische Darstellung beachten. Seine Arbeit konkurriert mit der des Web-Designers, in dessen Fokus eine für den Nutzer ergonomische Darstellung der Inhalte liegt. Da Anwendungscode und Darstellungselemente aber in ein und derselben Datei liegen, muss er sehr darauf achten, bei der Gestaltung der Seite nicht den Programmcode zu zerstören.

Die aktuellsten Ansätze versuchen, dieses Dilemma zu beheben. Die Architekturansätze der *Java 2 Enterprise Edition* sowie des .NET-Projekts von Microsoft trennen deutlich die Businesslogik von der Darstellung. Beide Ansätze bauen klar auf einer Drei-Schicht-Architektur auf. Kern beider Technologie-Plattformen ist ein Laufzeitsystem - die virtuelle Maschine (VM) - das einen neutralen Programmcode ausführt. Dies entkoppelt Programme von Hardware und Betriebssystem.

Beide Systeme weisen moderne Eigenschaften wie die Möglichkeit der Nutzung von Web-Services, Load-Balancing, Connection-Pooling von Datenbankverbindungen, asynchroner Kommunikation, etc. auf.

Allerdings wird die virtuelle Maschine von Microsoft – Common Language Runtime (CLR) – nur für Windows-Betriebssysteme angeboten [ST02]. Für Java dagegen existieren VMs für alle Betriebssysteme: von der Chipkarte bis zum Mainframe. Hinzu kommt, dass fast alle für die J2EE-Umgebung entwickelten Komponenten auch als open-source-Variante existieren. Die Betriebssystemunabhängigkeit und Kostenvorteile sowie eine beispielhafte Dokumentationslage

waren die Hauptgründe der Wahl der J2EE-Architektur mit dem Einsatz des Java-Application-Servers als Grundlage der vorliegenden Arbeit.

3.1.1.3 Das Datenbanksystem: Microsoft SQL Server

Das Datenbanksystem Microsoft SQL Server hat in der Version 2000 ein integriertes Data Warehouse. Insbesondere dieses Merkmal unterscheidet es von vergleichbaren Produkten. Da die Anbindung eines Data Warehouses ein in der vorliegenden Arbeit zu untersuchender Aspekt der Nutzung von Application-Servern ist, bot sich das Produkt an. Hinzu kommt, dass es an der Fakultät für Informatik bereits eingesetzt wird und man seinen hohen Integrationsgrad schätzt.

3.1.2 Eingesetzte Werkzeuge und Tools für die Entwicklung der Beschaffungsanwendung

Die nachfolgend genannten Bestandteile der Entwicklungsumgebung waren zum Zeitpunkt der Entwicklung auf einem Rechner installiert. Gegen Ende des Absatzes wird kurz geschildert, dass insbesondere der auf Schicht 2 eingesetzte Web-Container und das in Schicht 3 genutzte Datenbanksystem auch verteilt auf verschiedene Computer miteinander kommunizieren können.

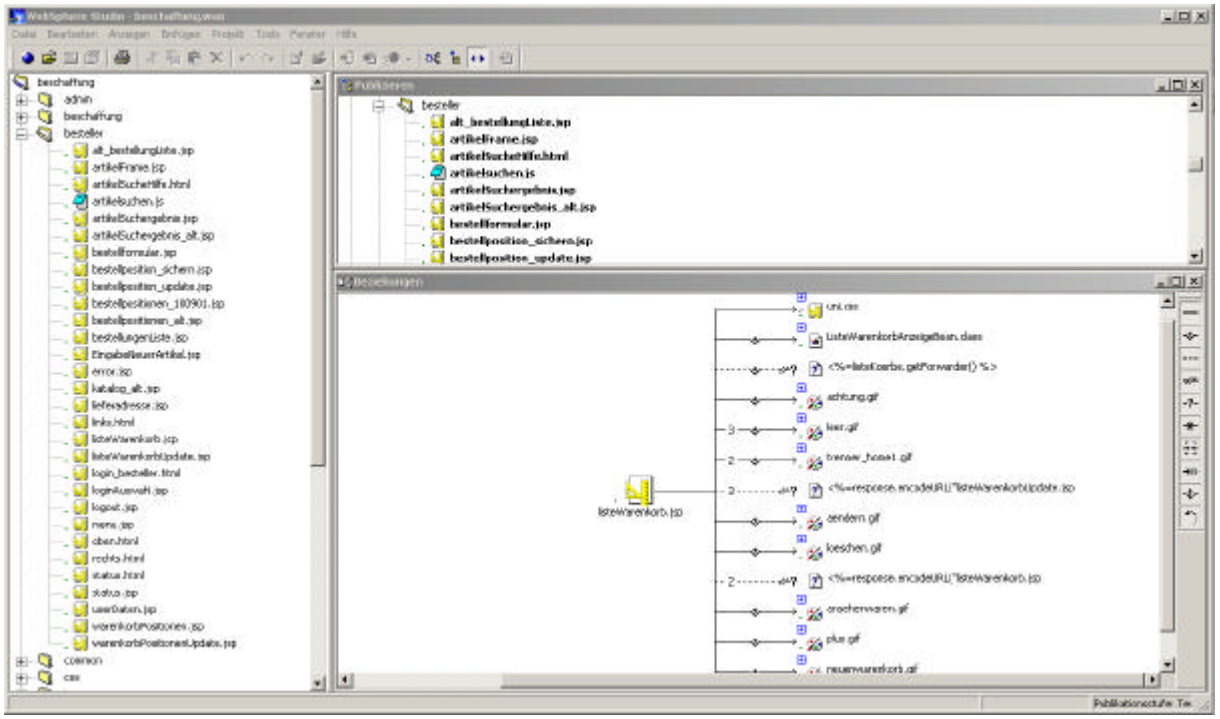


Abbildung 3-2: Websphere Studio, Version 4. Das linke Fenster zeigt den Strukturbaum des Projekts, das obere rechte Fenster die Publikationsstruktur, das heißt die Pfade innerhalb der Anwendungshierarchie des Application-Servers, in welche die erstellten HTML-Seiten und Java-Klassen übertragen werden. Das rechte untere Fenster zeigt ausgehend von einer Datei an, welche anderen Dateien sie referenziert, bzw. von welchen anderen Dateien sie refernziert wird.

Als Entwicklungsumgebung wurde ein Produkt aus der Websphere-Familie gewählt: Websphere Studio. Es ist ein Werkzeug zur Verwaltung eines Webprojekts und beinhaltet ebenfalls Komponenten für das Design von Internetseiten sowie zum Editieren von Java-Programmen. Obwohl es Teil der Websphere-Familie ist, kann Websphere Studio so konfiguriert werden, dass es die von Sun Microsystems definierte Struktur einer Application-Server basierten Web-Anwendung unterstützt. Neben den in der obenstehenden Abbildung skizzierten Eigenschaften kann man im Websphere Studio durch die Wahl einer Publikationsstufe schnell zwischen den Servern, auf die ein Projekt publiziert werden soll, umgeschalten. Sinnvoll ist dies bei der Nutzung von verschiedenen Servern für den Test- und Produktiveinsatz. Die Arbeit in Teams wird durch ein nutzerbasiertes Verwaltungssystem der entnommenen Dateien unterstützt. Es blockiert von einem Nutzer A zur Bearbeitung entnommene Dateien zur Änderung durch einen Nutzer B. Es beinhaltet in Form des Programms *Page Designer* auch ein Tool zur Erstellung von HTML- und JSP-Seiten.

Als Web-Container des Application-Servers (siehe Abbildung 3-1) wird in der Entwicklungsumgebung *Tomcat in der Version 3.3* eingesetzt, da diese Version das Production Release zum Zeitpunkt des Beginns der Entwicklung der Beschaffungsanwendung darstellt. Tomcat ist eine frei erhältliche Open-Source-Implementierung der Java Servlet und JavaServer Pages Technologien. Die Entwicklung der Spezifikationen dieser Technologien unterliegt der Firma Sun Microsystems, welche Tomcat in die *Referenzimplementierung* der J2EE-Spezifikation integriert. Tomcat selbst wird innerhalb des sogenannten Jakarta-Projektes der Apache Software Foundation entwickelt, da Sun Microsystems die Referenzimplementierung an diese lizenziert hat. Im Gegenzug verpflichtete sich Apache zur permanenten Weiterentwicklung des Projektes in präziser Umsetzung der Spezifikationen. In Tomcat 3.3 sind die Java Servlets Spezifikationen der Version 2.2 und JavaServer Pages Spezifikationen der Version 1.1 umgesetzt [SV00, SP00].
Da in Tomcat 3.3.1 bereits ein vereinfachter Webserver integriert ist, der in der Standardkonfiguration auf Port 8080 http-Requests entgegennimmt, konnte in der Entwicklungsumgebung auf die zusätzliche Installation eines Webservers verzichtet werden.

Der als Datenbanksystem zum Einsatz kommende SQL Server 2000 der Firma Microsoft stellt nach der Installation einen ODBC-Treiber zur Verfügung, mit dem ein problemloser Zugriff auf die verwalteten Datenbanken möglich ist.

3.2 Konzept der Beschaffungsanwendung

Das Konzept der Beschaffungsanwendung orientiert sich an der in Abschnitt 1.2.2 beschriebenen Zielstellung des Sachgebiets 12 und den in Kapitel 2 durchgeführten Prozessanalysen. Der folgende Abschnitt 3.2.1 gibt einen Überblick des funktionsspezifischen Aufbaus der entwickelten Anwendung.

In 3.2.2 wird das Datenbankschema dargestellt und Beweggründe für die Definition seiner Struktur erläutert. Der Abschnitt 3.2.3 widmet sich der Fragestellung, mit welchem konzeptionellen Ansatz die Beschaffungsanwendung die definierten Anforderungen erfüllt.

3.2.1 Nutzergruppen und Funktionsbereiche

Die Beschaffungsanwendung unterscheidet drei Nutzergruppen, die Nutzer der Einrichtungen, also die Besteller, die Mitarbeiter des Sachgebiets 12, und die Administratoren des Systems. Die letzte Gruppe wird bei den folgenden Betrachtungen ausgenommen, da sie für die Schilderung der abgebildeten Prozesse nicht relevant ist.
Insgesamt bietet die Anwendung ihren Nutzern zehn Funktionsbereiche an. Während den Mitarbeitern des Sachgebiets 12 alle Funktionen zur Verfügung stehen, ist die Auswahl der Einrichtungen eingeschränkt. Die Abbildung 3-3 nennt die Funktionsbereiche und zeigt gleichzeitig, welche Auswahl den Einrichtungen zur Verfügung steht.

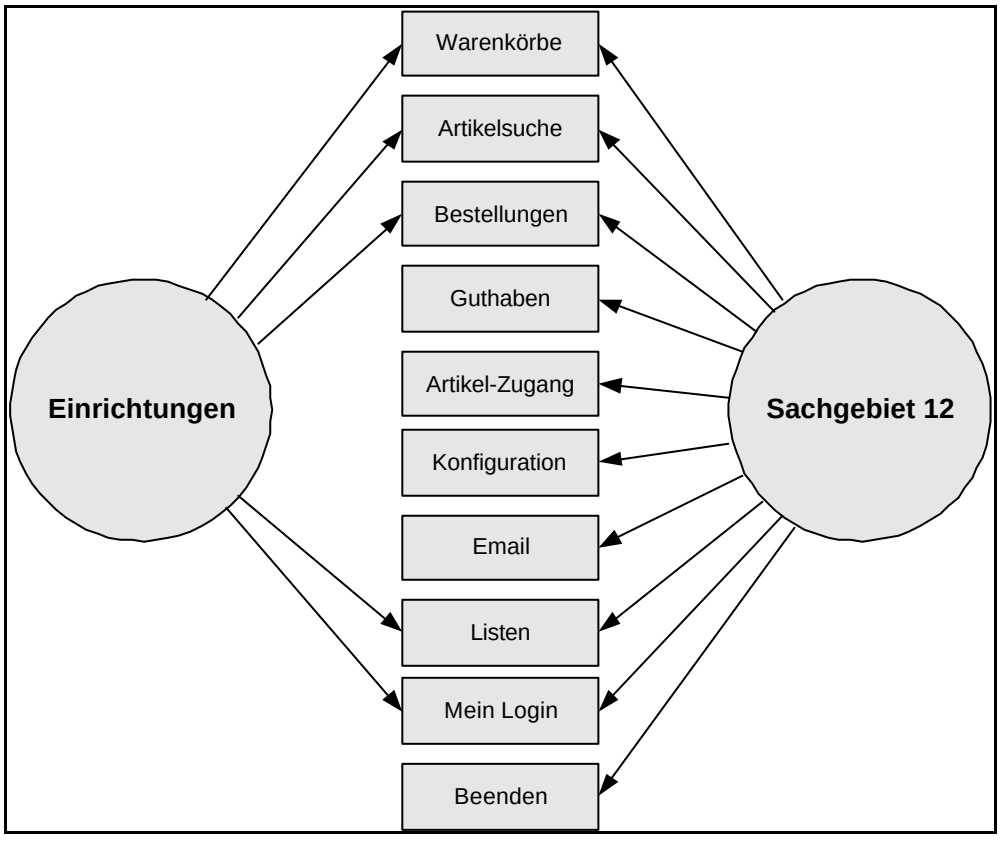


Abbildung 3-3: Darstellung der unterschiedlichen Funktionsbereiche, die von den Nutzern der Einrichtungen und des Sachgebiets 12 genutzt werden können

In dem Funktionsbereich *Warenkörbe* sind alle Funktionen subsummiert, die der Nutzer braucht, um eine Auswahl an benötigten Artikeln zusammenzustellen. Ein Nutzer kann parallel mehrere Warenkörbe füllen. Er bedient sich dabei der *Artikelsuche*, die ihm eine freie Suche über

Artikelbezeichnungen und –nummern erlaubt. Ist der Warenkorb zufriedenstellend gefüllt, bestellt er ihn. Er bekommt daraufhin eine Übersicht aller *Bestellungen*, die er oder andere Nutzer für die entsprechenden Kostenstellen bereits getätigt haben. Der Nutzer sieht in diesem Bereich den Status seiner Bestellungen und kann sich auch deren Positionen und den Wert der Bestellung anzeigen lassen. In den Wert einer Bestellung ist das *Guthaben* eingerechnet, das Kostenstellen für bestimmte Artikelgruppen ausschließlich von den Mitarbeitern des Sachgebiets 12 eingeräumt werden kann. Das Sachgebiet 12 hat weiterhin die Möglichkeit, *Artikel-Zugänge* zu buchen, um Menge und Preise der im Lager befindlichen Artikel zu aktualisieren. Unter *Konfiguration* haben die Mitarbeiter Zugriff auf die Verwaltung der Kostenstellen, Nutzerdaten und weitere Einstellungen. Die *Email*-Funktion ermöglicht ihnen, Nachrichten an bestimmte oder alle Nutzer der Beschaffungsanwendung zu schicken. *Listen* werden allen Nutzergruppen angeboten. Sie geben eine übersichtliche Darstellung der bestellten Artikel und Werte nach verschiedensten Kriterien. Unter *Mein Login* kann der jeweilige Nutzer seinen Nutzernamen und sein Passwort ändern. *Beenden* ermöglicht das Verlassen der Anwendung und deaktiviert die ihm zugeordnete Session.
Die geschilderten Funktionalitäten werden an späterer Stelle in diesem Kapitel im Rahmen der Diskussion, inwieweit sie die im Abschnitt 2.5 aufgeführten *Ansatzpunkte für die Prozessoptimierung* beantworten bzw. realisieren, intensiver beleuchtet.

3.2.2 Beschreibung des Datenbankschemas und seiner Motivation

Die Abbildung 3-4 zeigt das Datenbankschema der Beschaffungsanwendung. In der Mitte des Schemas befindet sich die Tabelle: *User_Stammdaten*, die mit sieben bezugnehmenden Entities eine zentrale Rolle spielt. Über die 1:1 – Beziehung mit der Tabelle *User_Login* wird gewährleistet, dass jedes Element aus der Entity User_Stammdaten, also ein Nutzer, genau einem ihn beim Anmeldevorgang authentifizierenden Element aus der Entity User_Login, also der Username/Passwort-Kombination zugeordnet ist. Die Entity *User_Type* beinhaltet die Information, welcher Gruppe der Nutzer abgehört: Besteller, Mitarbeiter des Sachgebiets 12 oder Administratoren. Da sowohl der Fall existiert, dass Nutzer für mehr als eine Kostenstellen bestellen dürfen, als auch der Fall, dass für bestimmte Kostenstellen von mehreren Nutzern Bestellungen ausgelöst werden können, existiert zwischen der Tabelle *User_Stammdaten* und *BestellerKostenstellen* eine m:n-Beziehung. Die Tabelle *BestellerKostenstellen* beinhaltet pro Nutzer also alle Kostenstellen, für die der jeweilige Nutzer bestellen darf. Sie ist verwandt mit der Tabelle *KostenstellenAusUni*, welche immer alle Kostenstellen beinhaltet, die an der Universität bekannt sind. Während diese Entity in den ersten Entwicklungsphasen durch eine aus dem HIS MBS exportierten Access-Tabelle aller bekannten Kostenstellen geladen wurde, bietet die Anwendung im vorstehend genannten *Funktionsbereich Konfiguration* einen Dialog an, mit dem Änderungen eingegeben werden. Die für die Vergabe im HIS MBS zuständigen Sachbearbeiter geben Änderungsinformationen jeweils an Mitarbeiter des

Sachgebiets 12 weiter. Leider konnte aufgrund des fehlenden Zugangs zu dem Buchungsmodul von HIS MBS keine automatische Synchronisation der Daten realisiert werden.

Sehr ähnlichen Charakter haben die Beziehungen der Tabelle *User_Stammdaten* zu den Entities *Bestellung, Lieferung* und *Warenkorb.* Nutzer können jeweils beliebig viele Elemente dieser Entities anlegen. Weitere Analogien gibt es in den Beziehungspaaren *Bestellung-BestellungPositionen*, *Lieferung-LieferungenPositionen* und *Warenkorb-WarenkorbPositionen.* Bei den in jedem Paar als Zweiten genannten, handelt es sich jeweils um schwache Entities.

Hervorzuheben ist die rekursive Relation der Entity *BestellungPositionen.* Diese Relationship-Menge repräsentiert den Fall, dass die Bestellposition einer Bestellung von den Mitarbeitern des Sachgebiets 12 geändert wurde, bspw. in der Bestellmenge, dem Preis oder sogar insgesamt, das heißt sie wurde gegen einen Ersatzartikel ausgetauscht. In diesen Fällen wird der Vorgänger jeweils erhalten und angezeigt, damit die Besteller nachvollziehen könne, welche Änderung vorgenommen wurde.

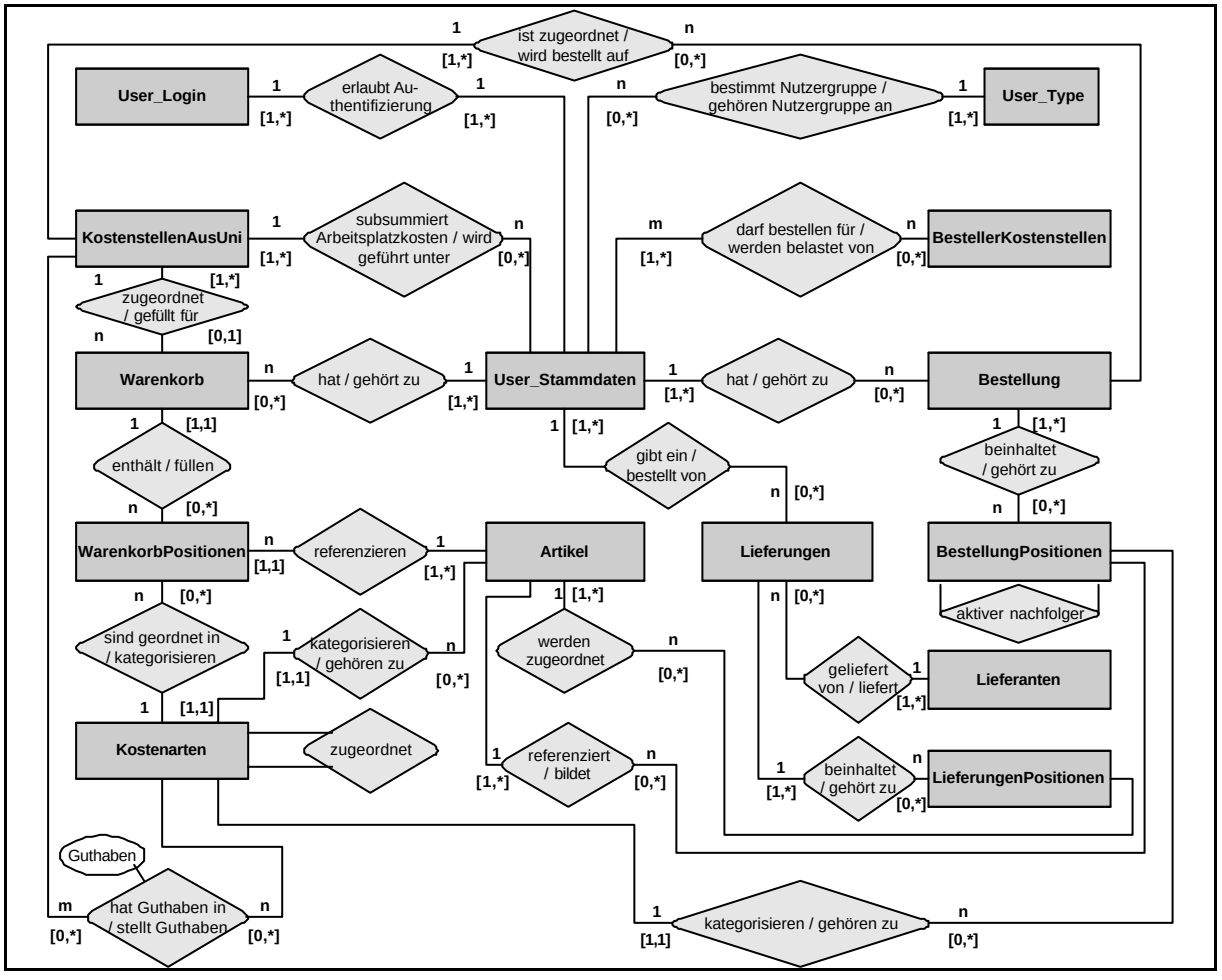


Abbildung 3-4: Datenbankschema der Beschaffungsanwendung.

Bei der Relation der vorstehend genannten drei schwachen Entities zu der Entity Artikel gibt es insofern eine Besonderheit, als das große Teile der Attribute der Tabelle Artikel in den genannten

Tabellen redundant gehalten werden. Die Gründe dafür sind unterschiedlich. In der Tabelle *LieferungenPositionen* ist die Redundanz der Felder Bezeichnung (Bezeichnung des Artikels) oder Einheit (Einheit der Menge, die geliefert wurde) notwendig, um die vom Lieferanten möglicherweise abweichend formulierten Artikelbezeichnungen ablegen zu können. Gleichzeitig wird mit dem Foreign-Key *ID_Artikel* aber auf den korrespondierenden Eintrag der Tabelle *Artikel* verwiesen. Etwas komplizierter ist die Argumentation im Fall der Entity *WarenkorbPositionen*. Hier werden die redundant einen Artikel beschreibenden Attribute nur dann gefüllt, wenn es sich um einen Artikel handelt, der nicht im Artikelstamm vorhanden ist. Andernfalls wird nur die Bestellmenge und der Foreign-Key als Verweis des in den Warenkorb gelegten Artikels gespeichert. Dies führt dazu, dass zur Summenberechnung des Warenkob-Wertes sowohl das Produkt aus Menge und Wert der in der Tabelle *WarenkorbPositionen* vollständig enthaltenen (also neuen) Artikel herangezogen werden muss, als auch die Bestellwerte der referenzierten Artikel. Durch dieses Vorgehen wird erreicht, dass sich Änderungen in den Attributen eines Artikels sofort auf die Artikelangaben im Warenkorb übertragen. Wird ein Warenkorb bestellt, werden die Artikeldaten in die Bestellpositionen kopiert und damit fixiert. Die Entity *Kostenarten* subsummiert Artikel in Gruppen. Sie hat Relationen zu allen Entities, in denen Artikel referenziert werden. Sie ordnet die Artikel jeweils einer Kostenart zu. Obwohl zu Beginn der Erstellung des Schemas nur eine Verbindung zu der Entity *Artikel* notwendig schien, zeigte sich, dass die häufiger auftretenden Änderungen in der Zuordnung von Artikeln zu Kostenarten, also Artikelgruppen die weiteren Relationen notwendig machten, um entscheiden zu können, in welcher Kostenart sich bspw. ein bestellter Artikel *zum Zeitpunkt* der Bestellung befand.
Zwischen den Enities *KostenstellenAusUni* und *Kostenarten* existiert eine m:n-Relation mit dem Attribut *Guthaben.* Die Mitarbeiter des Sachgebiets 12 haben somit die Möglichkeit, jeder Kostenstelle ein Guthaben für eine bestimmte Gruppe von Artikeln (*Kostenarten*) zuzuordnen.
Die Entity *KostenstellenAusUni* steht ebenfalls mit der Entity *Bestellung* in Relation. Die Relation ordnet jeder Bestellung die Kostenstelle zu, auf die sie gebucht werden soll.

3.2.3 Konzeption der Beschaffungsanwendung im Hinblick auf gestellte Anforderungen

Betrachtet man die aus der Prozessanalyse gefolgerten Forderungen für das entwickelte System, so lassen sich vier Hauptbereiche ausmachen (Abbildung 3-5).

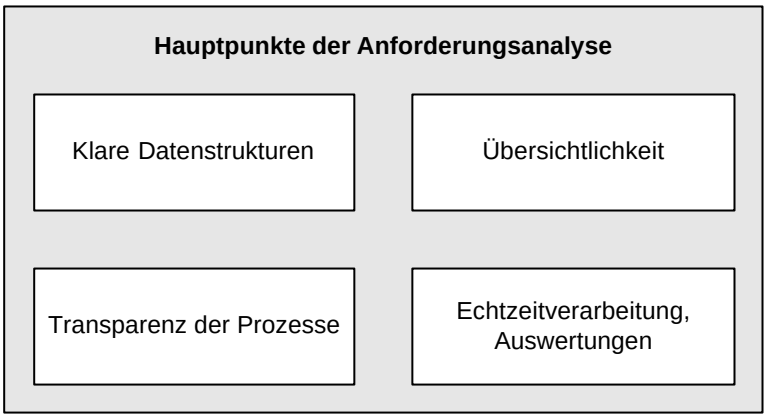


Abbildung 3-5: Die vier Hauptpunkte der Anforderungsanalyse.

Klare Datenstrukturen

Die Definition der Datenstrukturen ist ein zentraler Punkt bei der Entwicklung der Anwendung. Eine Übersicht dazu gibt das Datenbankschema. Besondere Auswirkungen auf das Konzept der Beschaffungsanwendung haben die mit der Einführung der Kosten- und Leistungsrechnung Einzug haltenden Kostenarten. Kostenarten sind in Bezug auf die entwickelte Applikation als die Gruppierung von Artikeln zu verstehen (siehe Abschnitt 2.1). Die Kostenarten geben der Universität Leipzig die Möglichkeit, Aussagen über den Verbrauch verschiedenster Güter und Dienstleistungen, heruntergebrochen bis auf einzelne Kostenstellen zu treffen.

Während des Abstimmungsprozesses mit dem Sachgebiet 12 wurde deutlich, dass sich die aus dem Artikelkatalog bekannte Gruppierung mit den definierten Artikelgruppen nur unzureichend abbilden lässt. Der Artikelkatalog beinhaltete weit mehr Artikelgruppen. Sie dienten den Mitarbeitern des Sachgebiets 12 zur einfacheren Zuordnung von Lagerplätzen. Innerhalb der Beschaffungsanwendung wurde deshalb die Möglichkeit geschaffen, beliebig viele Artikelgruppen anzulegen und diese hierarchisch anzuordnen. Durch die notwendige rekursive Relation der Tabelle Kostenarten kennt jede Artikelgruppe ihre übergeordnete Gruppe. Diese Struktur hat mehrere Vorteile. Sie erlaubt eine sehr flexible Einführung von neuen Artikelgruppe und die Verschiebung ganzer Unterbäume. Jeder Artikel hat einen Foreign-Key der Entity Kostenarten und verweist damit auf die Artikelgruppe, der er zugeordnet wird. Die Nummer eines Artikels besteht in der Ansicht für den Nutzer aus der sechsstelligen Nummer der Kostenart, der er zugeordnet ist und einer eigenen vierstelligen Nummer.

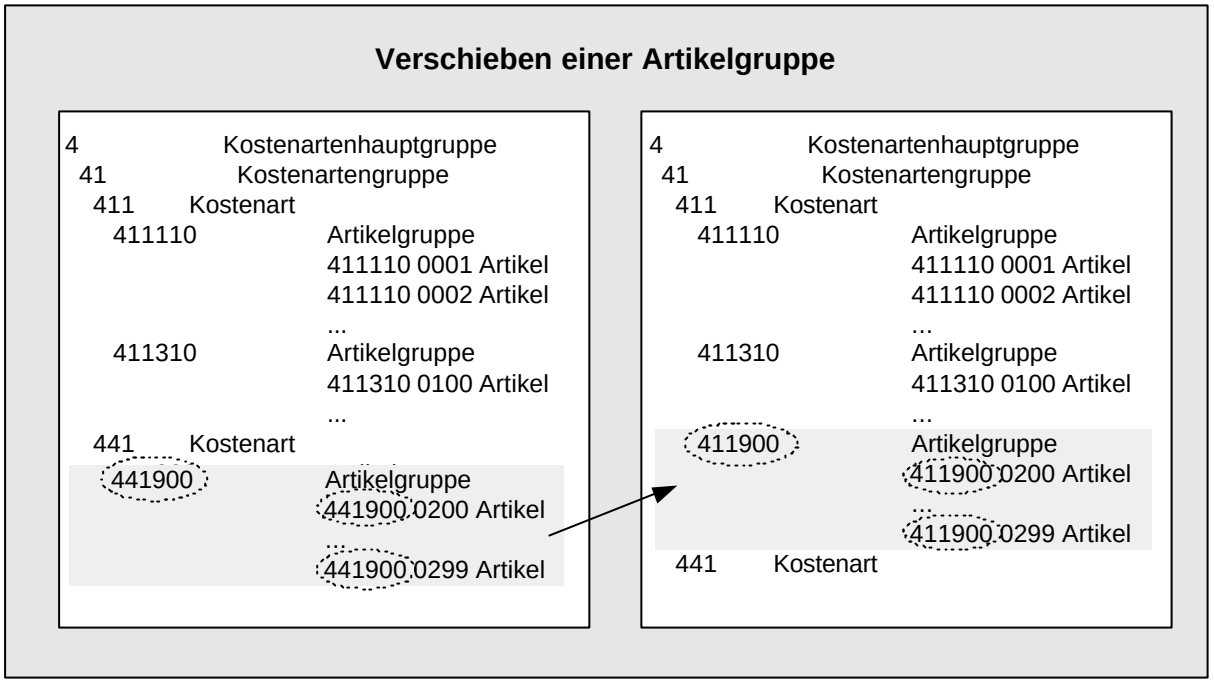


Abbildung 3-6: Verschieben einer Artikelgruppe. Durch das Verschieben werden auch alle der Gruppe untergeordneten Artikel verschoben und die ersten sechs Stellen der Artikelnummer angepasst (siehe ovale Markierungen).

Weil die Definition der Kostenarten durch die Universität Leipzig noch nicht abgeschlossen ist, gab es bereits einige Fälle, in denen Kostenarten umsortiert werden mussten. Das vorstehend erläuterte Zusammenspiel von Kostenarten und Artikelstamm bildet diese Änderungen effizient nach.

Die vorstehend geschilderte Struktur bietet eine klare Regelung der vergebenen Artikelnummern und hebt die Trennung von Bestellungen des Bewirtschaftungsbedarfs und Bürobedarfs auf. Eine weitere Forderung der Datenanalyse war mit dem Punkt B des Abschnitts 2.5 die Einführung einer Strategie zur Einschränkung von Zugriffsmöglichkeiten auf die verwalteten Datenbestände und angebotenen Funktionalitäten. Wie bereits weiter oben kurz skizziert, wird in der Beschaffungsanwendung jedem Nutzer eine eindeutige Rolle, in der er sich im System bewegt, zugeordnet. Angelegt sind die Rollen: *Besteller* (Nutzer der bestellenden Einrichtungen), *Beschaffer* (Mitarbeiter des Sachgebiets 12) und *Administratoren*. Die beiden letztgenannten haben identische Zugriffsmöglichkeiten. Die Rolle *Administratoren* ist für zukünftige Anwendungszwecke ausgelegt.
Die Rollenbezeichnungen werden in der Datenbankstruktur in der Entity User_Type geführt. Welcher Rolle ein Nutzer angehört, wird nicht bei jedem Zugriff auf Funktionalitäten der Anwendung geprüft sondern einmalig beim Einloggen des Nutzers in das System. Die ihm zugeordnete Rolle wird nach erfolgreichem Login-Vorgang in Form einer Session-Variable gespeichert. Dies erspart kostenintensive Zugriffe auf das Datenbanksystem. Die Rolle eines Nutzers definiert die Funktionalitäten, die ihm in Form des Menüsystems (siehe Abbildung 3-3), aber auch auf den

Inhaltsseiten angezeigt werden. So sehen beispielsweise nur die Angehörigen der Rolle *Beschaffer* einen Button, der es ihnen ermöglicht, Bezeichnungen in der Artikelliste zu ändern.
Während alle Mitarbeiter des Sachgebiets 12 Zugriff auf alle Bestellungen haben, egal auf welche Kostenstelle sie ausgelöst wurden, sind die Zugriffsmöglichkeiten der Nutzer von Einrichtungen beschränkt. Sie können nur für solche Kostenstellen bestellen, die ihnen in der Entity *BestellerKostenstellen* zugeordnet wurden (siehe dazu auch die Erläuterung des Datenbankschemas). Die Zuordnung der Kostenstellen zu Nutzern wird vom Sachgebiet 12 verwaltet. Folgerichtig können sich die Besteller auch nur Auswertungen der Kostenstellen ansehen, für die sie als bestellberechtigt eingetragen wurden. Eine Ausnahme existiert insofern, als dass es in den Einrichtungen fast immer bestimmte Nutzer gibt, die einen Überblick über die insgesamt für eine gesamte Fakultät ausgelösten Bestellungen haben müssen. Das Sachgebiet 12 kann diesen Nutzern individuell das Recht geben, Auswertungen über eine gesamte Fakultät zu generieren.

Transparenz der Prozesse

Die fehlende Transparenz der Bestellprozesse hat verschiedene Ursachen. Einerseits ist die Einführung eines für alle Benutzer zugänglichen Beschaffungssystems an sich ein Schritt zu mehr Transparenz, weil sie die Basis für die Echtzeitanzeige der Bestellprozesse ist.
Die Analyse zeigt aber auch, dass mit der Forderung nach Transparenz in erster Linie gemeint ist, nachvollziehbar zu machen, in welchem Prozessschritt sich eine Bestellung befindet. In der Beschaffungsanwendung wird der Prozessschritt durch die Einführung eines Statuswerts abgebildet.

Mögliche Statuswerte sind:

- *Neue Bestellung* bezeichnet eine Bestellung, die noch nicht vom Sachgebiet 12 bearbeitet wurde.
- *In Arbeit* bezeichnet eine Bestellung, die vom Sachgebiet 12 bereits bearbeitet wird.
- *Fertiggestellt* bezeichnet eine Bestellung, die vom Sachgebiet 12 fertiggestellt ist und entweder schon geliefert oder zur Auslieferung bereitgestellt wurde.
- *Storno angefordert* bezeichnet eine Bestellung, die von der bestellenden Einrichtung storniert wurde, der Storno ist vom Sachgebiet 12 noch nicht bestätigt.
- *Storniert* bezeichnet eine Bestellung, die vom Sachgebiet 12 storniert wurde.
- *Buchungsbereit* bezeichnet eine Bestellung, die von der Einrichtung zum Buchen freigegeben ist.
- *Gebucht* bezeichnet eine Bestellung, die an eine externes Buchungssystem weitergereicht wurde.

Die Abbildung 3-7 zeigt die häufigsten Pfade der Statuswechsel.

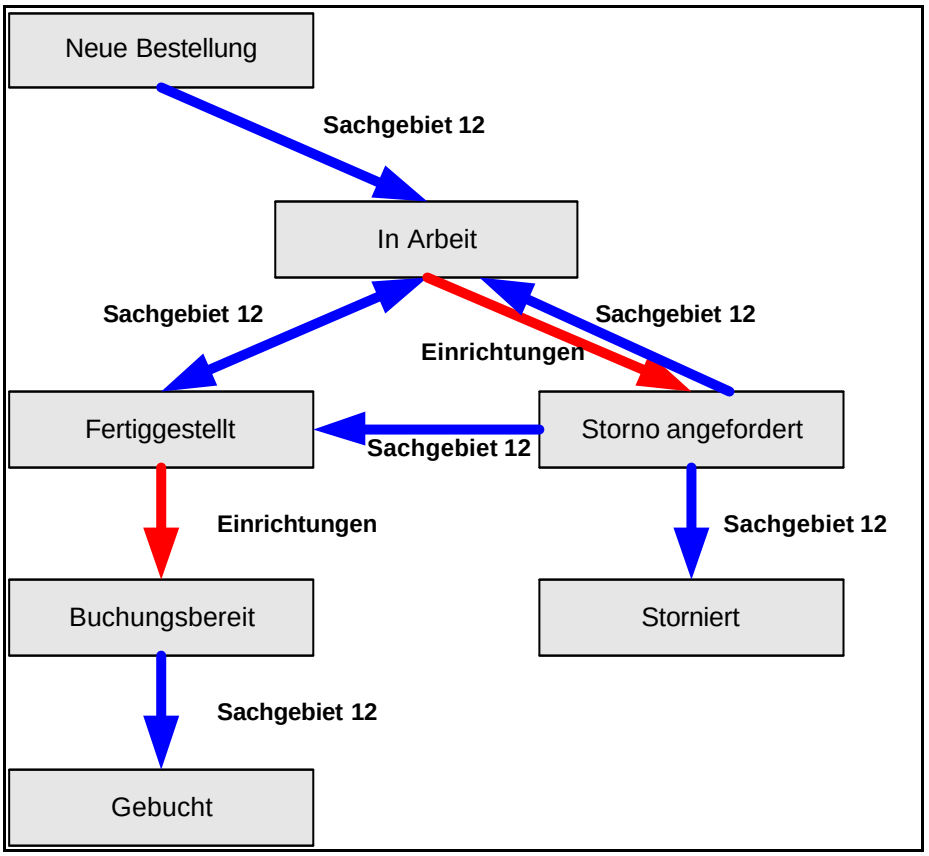


Abbildung 3-7: Darstellung der möglichen Statuswechsel

Wenn der Nutzer einer Einrichtung eine Bestellung auslöst, bekommt diese den initialen Statuswert: *Neue Bestellung*. Sobald die Mitarbeiter des Sachgebiets 12 die Bestellung bearbeiten, setzen sie den Statuswert auf *In Arbeit*. Die Einrichtungen wissen auf diese Weise, dass ihre Bestellung vom Sachgebiet 12 in den Beschaffungsprozess aufgenommen wurde. Sollte sich die Einrichtung entscheiden, die Bestellung zu stornieren, kann sie eine Stornoanforderung beantragen, indem sie den Status auf *Storno angefordert* setzt. Der eigentliche Storno kann nur durch das Sachgebiet 12 durchgeführt werden, indem das Sachgebiet den Status auf *Storno* setzt.
Wird von der bestellenden Einrichtung kein Storno angefordert, setzen die Mitarbeiter des Sachgebiets 12 den Status auf *Fertiggestellt*, sobald die Bestellung ausgeliefert werden kann.
Sind die bestellten Artikel geliefert worden, setzt die Einrichtung den Status auf *Buchungsbereit* und akzeptiert somit die Lieferung. Der Status *Gebucht* existiert für den Fall, dass eine Bestellung an ein externes Buchungssystem weitergereicht wurde.

Das dargelegte Statussystem ist ein Hauptpfeiler der Beschaffungsanwendung. Als solcher realisiert er auch die im Bereich der Transparenz analysierten Forderungen.

Übersichtlichkeit des Nutzerinterface der Anwendung

Die Übersichtlichkeit wird in der Beschaffungsanwendung durch verschiedene Eigenschaften erreicht. Einerseits werden die verfügbaren Funktionsbereiche deutlich in Form des weiter oben geschilderten Menüsystems unterschieden. Die Navigation über möglichst wenige, aber klar bezeichnete Buttons und Grafiken soll dem Nutzer helfen, Warenkörbe mit Artikeln zu füllen und zu bestellen. Die Suchfunktion für den Artikelkatalog ist möglichst einfach gehalten, beinhaltet aber alle notwendigen Hilfsmittel. Das Design ist stark an die Richtlinien des Webauftritts der Universität Leipzig angepasst.

Einen Überblick über diesen Bereich bietet der nächste Bereich. Er gibt im Rahmen der Implementierung anhand von Screenshots der Anwendung Beispiele für die oben aufgezählten Elemente.

Echtzeitzugriff auf Daten und bereitgestellte Auswertungen

Die Forderung nach einem Echtzeitzugriff auf die Daten der Beschaffungsanwendung ist leicht zu erfüllen, da er der gewählten Architektur immanent ist. Dies betrifft die Aktualisierung der Artikeldaten genauso wie das Auslösen einer Bestellung, die Änderung einer Kostenartenkonfiguration oder das Generieren von Auswertungslisten. In allen Fällen wirken sich die Änderungen in Echtzeit auf die verknüpften Daten aus und werden unmittelbar in zeitlich nachgelagerten Zugriffen aktualisiert angezeigt.

3.2.4 Betrieb der Beschaffungsanwendung im ASP-Modell

Ein wichtiger Gesichtspunkt des Einsatzes der Beschaffungsanwendung ist neben der Entwicklung auch die weitere Pflege und die Sicherung der Betriebsbereitschaft. Das Dezernat 1 der Universität Leipzig entschied sich dafür, die Anwendung auf den Servern eines externen Dienstleisters hosten zu lassen. Der Dienstleister ist ebenfalls verantwortlich für die Weiterentwicklung der Anwendung, also die Entwicklungen, die über die in der vorliegenden Arbeit vorgestellten Funktionsumfang herausgehen. Gründe für den Betrieb der Lösung im Application-Service-Providing (ASP) – Modell ist auf technischer Seite das fehlende KnowHow bei dem Betrieb von javabasierten Application-Servern seitens des Rechenzentrums der Universität sowie die entstehenden Kosten für den Wartungs- und Pflegeaufwand der Server und Anwendung. Insbesondere das von dem gewählten Dienstleister angebotene Paket aus Hosting- und Wartungskosten zusammen mit einem definierten, monatlich abrufbaren Aufwand für Weiterentwicklungen erwies sich nach internen Kalkulationen des Dezernats 1 als kostengünstiger und flexibler als der Eigenbetrieb.

3.3 Implementierung

Die Funktionalitäten der Beschaffungsanwendung sind seitens des Application-Servers unter Nutzung von JavaServer Pages und JavaBeans realisiert. Auf die Datenbasis greifen diese Java-Klassen unter Nutzung verschiedener Mechanismen zu. Der folgende Abschnitt beschreibt das Zusammenspiel zwischen den Java-Programmen untereinander und mit der Datenbank.
Anschließend werden detailliert die von der Beschaffungsanwendung angebotenen Funktionalitäten beschrieben.

3.3.1 Implementierungsmodell

Die Implementierung der Beschaffungsanwendung nutzt auf Ebene des Application-Servers JavaServer Pages (JSP) zur Präsentation von Inhalten und JavaBeans als Träger der Business-Logik. Um das Zusammenspiel zwischen diesen beiden J2EE-Bestandteilen und ihrem Bezug zum Servlet-Container zu demonstrieren, wird in der folgenden Abbildung die mittlere Schicht der verwendeten Architektur fokussiert.

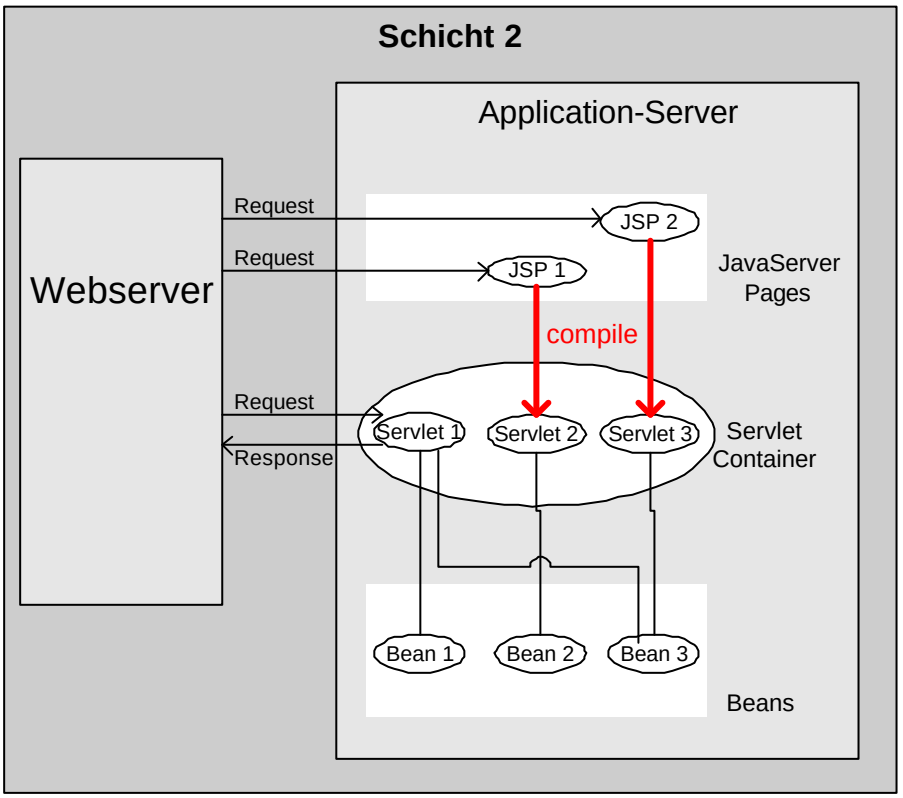


Abbildung 3-8: Aufrufe von Servlets und JavaServer Pages unter Nutzung von JavaBeans im Application-Server

Der Webserver reicht sowohl *Servlet-Aufrufe* als auch Aufrufe von *JavaServer Pages* an den Application-Server weiter. Der Servlet-Container kann Servlets unmittelbar ausführen und an den

Webserver zurückschicken. Der Aufruf von JavaServer Pages erfordert erst einen Kompiliervorgang, bei dem JSPs in Servlets umgewandelt werden, damit der Servlet-Container sie ausführen kann. Der Kompiliervorgang wird mit Hilfe der installierten Java-Umgebung und einer vom Application-Server gelieferten Klassen-Erweiterung realisiert. Servlets, und damit auch JSPs, können Beans instantiieren. Ein Bean stellt dem Servlet bzw. der JavaServer Page in gekapselter Form Business-Logik zur Verfügung. Wie die Abbildung 3-8 zeigt, können Servlets auch mehr als ein Bean einbinden.

Die Einbindung der Beans erfolgt in der Beschaffungsanwendung über die Nutzung der Standardaktion <jsp:useBean:>. Aktionen sind spezifische Tags, die sich auf das Laufzeitverhalten einer JSP-Seite und auf die Antwort auswirken, die zum Client zurückgesendet wird. Die JSP-Spezifikation führt einige Standardaktionen auf. Diese müssen ungeachtet der Implementierung von allen Containern zur Verfügung gestellt werden. Standardaktionen versorgen die JSPs mit einigen Grundfunktionen. Das nachfolgende Quelltextfragment zeigt die Einbindung eines Beans am Beispiel der JSP-Date: *listeWarenkorb.jsp*. Sie ist zuständig für die Anzeige der Warenkörbe, die man in der Anwendung bspw. nach dem Klicken auf den Menüpunkt: *Warenkörbe* erhält.

```
<jsp:useBean       id="listeKoerbe"
                   class="besteller.ListeWarenkorbAnzeigeBean"
                   scope="request">
</jsp:useBean>
<% listeKoerbe.initBean(request); %>
```

Abbildung 3-9. Quelltext-Fragment der JavaServer Page *listeWarenkob.jsp*. Sie zeigt die Einbindung eines Beans, die Übergabe von Form-Objekten mit Hilfe der *setProperty*-Aktion und die Ausführung der Methode *initBean()* des Beans

Die Aktion *jsp:useBean* verknüpft ein JavaBean mit der JSP-Seite, welche die Aktion enthält. Sie gewährleistet, dass das Objekt für den im Tag bestimmten Gültigkeitsbereich zur Verfügung steht. Das gebundene Objekt kann in der JSP-Seite (oder abhängig vom Gültigkeitsbereich in anderen JSP-Seiten) über die mit dem Objekt verknüpfte ID referenziert werden.
Im oben stehenden Beispiel wird die Methode initBean über die vorher definierte *id listeKoerbe* referenziert. Der Parameter *class* gibt dem Container die Information, welche Java-Klasse als Bean instantiiert werden muss, der Parameter *scope* schränkt den Gültigkeitsbereich des Beans auf den request des JSP ein. Der Gültigkeitsbereich kann erweitert werden. So ermöglicht die Angabe *session* als Wert des Parameters *scope* auch anderen JSPs den Zugriff auf das einmal instantiierte Bean, solange sie in der selben Session aufgerufen werden.

Beans kapseln business-logische Algorithmen und bieten einem JSP in Form von öffentlichen Methoden Zugriff auf die Initialisierung von Anfangswerten und die Resultate der beaninternen Verarbeitung. In der Beschaffungsanwendung werden alle Beans von der Klasse: *BeanTemplateV1* abgeleitet. Sie implementiert einige Objekte und Methoden, die in fast allen Beans und den auf sie zugreifenden JSPs genutzt werden.

Dazu gehört beispielsweise das Objekt *UserMessage*, das genutzt wird, um während der Abarbeitung der Business-Logik Nachrichten an den User zu speichern. Der typische Ablauf der JSP-Bean – Verarbeitung sieht so aus, dass unmittelbar nach der Einbindung des Beans eine Initialisierungsmethode aufgerufen wird (siehe Abbildung 3-9). Sie wertet an das JSP eventuell übergebene Parameter aus und bereitet die Daten auf, die vom JSP angezeigt werden sollen. Ergibt sich während der Verarbeitung im Bean eine Situation, die es erfordert, dem Nutzer eine Nachricht zukommen zu lassen, füllt das Bean das Vector-Objekt UserMessage. Sowohl dieses Objekt als auch die es füllende Methode sind in der vererbenden Klasse BeanTemplateV1 *protected* gesetzt, so dass es im Bean einen privaten, also einzig auf die Beanklasse beschränkten Gültigkeitsbereich hat. Somit ist auch umgangen, dass eventuell die das Bean als Objekt instantiierende JSP-Datei einen direkten schreibenden Zugriff auf das vector-Objekt erhält. Der lesende Zugriff ist möglich, da die entsprechende Methode getUserMessage() in *BeanTemplateV1* mit dem Gültigkeitsbereich public definiert wurde.

In der Klasse *BeanTemplateV1* sind auch Methoden und Objekte definiert, die einen Datenbankzugriff unterstützen. In den meisten Fällen wird jedoch eine eigene Klasse entworfen, der die Verbindung zu einer Entity herstellt. Innerhalb dieser Zugriffsklassen werden die Tabellen oder Views fast ausschließlich durch die Ausführung der im Microsoft SQL Server angebotenen *Stored Procedures* realisiert. Ihre Handhabung ist sehr einfach, und erlaubt die Nutzung der vielfältigen Möglichkeiten des Datenbanksystems. Insbesondere das Cursor- und Transaktionskonzept bietet sich zur Nutzung an. Die folgenden Codefragmente demonstrieren die Ausführung einer *Stored Procedure* aus einer vom Bean instantiierten Zugriffsklasse.

```
CREATE PROCEDURE getGroupPositionenByBestellung
@id_bestellung int=null

AS


IF (@id_bestellung=null)
     SELECT *
     FROM groupPositionenByBestellung
ELSE
     SELECT *
     FROM groupPositionenByBestellung
     WHERE ID_Bestellung=@id_bestellung
GO
```

Abbildung 3-10: Die Stored Procedure getGroupPositionenBestellung führt in Abhängigkeit des übergebenen Parameters zwei verschiedene SELECT-Anweisungen aus.

```
newDataBase("BestellungPositionen");
...
execute("BestellungPositionen", "getGroupPositionenByBestellung "+id);
String wert = nextExecuteRow("BestellungPositionen");
if (wert != null) {
       String Bez = getString("BestellungPositionen", "Bezeichnung")
}
```

Abbildung 3-11: Codefragmente der Datei BestellungPositionen.java, in die Stored Procedure BestellungPositionen aufgerufen wird.

Die Datenbankzugriffsmethoden werden der Klasse BestellungPositionen durch Vererbung zur Verfügung gestellt. Mit der Methode *newDataBase(String ConnectionName)* wird eine neue Connection zur Datenbank erzeugt. Mit der Methode *execute(String, String)* führt im weiteren Ablauf über diese Connection eine Stored Procedure mit dem Parameter id aus. Die Stored Procedure prüft, ob das Feld id einen Wert hatte oder auf Null gesetzt war und führt in Abhängigkeit des Ergebnisses zwei verschiedene SELECT-Anweisungen aus. Die Methode *nextExecuteRow(String)* liest den jeweils nächsten Datensatz des *ResultSets* der Stored Procedure aus. Die einzelnen Feldinhalte können dann mit der Methode *getString(String, String)* ermittelt werden, wobei der erste String wie bei allen vorstehend genannten datenbankbezogenen Anweisungen den ConnectionName spezifiziert.
Ein Grossteil der hinter diesen Methoden stehenden Programmlogik musste neu implementiert werden. Insbesondere die Methode *nextExecuteRow(String)* ist von hoher Komplexität. Obwohl der in Klassen DataBaseAccessV2 definierte Code in der aktuellen Version nur knapp 40 Zeilen Code enthält, ist das Zusammenspiel und die gegenseitigen Abhängigkeiten der verschiedenen Statuswerte des Statement-Objekts, das die Verbindung zur Datenbank herstellt, nicht offensichtlich. Die Transparenz der Datenbankzugriffsmethodik wird unter anderem dadurch reduziert, dass in der Implementierung des vom SQL Server installierten ODBC-Treibers ein Bug versteckt liegt. Er tritt zutage, wenn eine Stored Procedure mehrere ResultSets verschiedener Struktur zurückliefert. Da die Metadaten des zweiten ResultSets nicht korrekt aktualisiert werden, kann die getString-Methode nicht auf die Attribute des zweiten ResultSets zugreifen.

Die vorstehend dargestellten Implementierungsmodelle reflektieren den wichtigsten Teil der in der Bestellungsanwendung eingesetzten Mechanismen. Mit den vorgestellten Modellen sollte es möglich sein, die nachstehende detaillierte Beschreibung der Beschaffungsanwendung aus Implementierungsgesichtspunkten nachzuvollziehen.

3.3.2 Detaillierte Beschreibung

Die folgenden Abschnitte behandeln die vorstehend genannten Funktionsbereiche. Pro Bereich werden die implementierten Funktionalitäten beschrieben und in Form von ein bis zwei Screenshots

visualisiert. Des weiteren werden die entwickelten JavaServer Pages genannt und erläutert sowie in Form eines Diagramms den Java Beans zugeordnet, in denen die jeweilige Business-Logik codiert ist. Das Diagramm kombiniert diese Zuordnung mit einer schematischen Darstellung der Vererbungshierarchie der Java Beans.
Die Screenshots zeigen auch das Design der Anwendung. Es orientiert sich des eng an den vom Dezernats 6 der Universität Leipzig im ersten Quartal 2002 spezifizierten Gestaltungsrichtlinien zum Aufbau von Internetseiten.

3.3.2.1 Funktionsbereich Warenkörbe

Unter dem Menüpunkt Warenkörbe ist das Anlegen von Warenkörben sowie ihre Verwaltung zusammengefasst. In der Beschaffungsanwendung trägt jeder Warenkorb eine Bezeichnung und ist einer Kostenstelle zugeordnet. Die Namensgebung ist als Orientierung für den Nutzer insofern sinnvoll, als dass er mehr als einen Warenkorb parallel füllen kann. Sowohl Bezeichnung als auch Kostenstelle können solange verändert werden, bis der Warenkorb bestellt wird. Änderungen in diesen Attributen haben keine Auswirkung auf die im Warenkorb befindlichen Artikelpositionen.

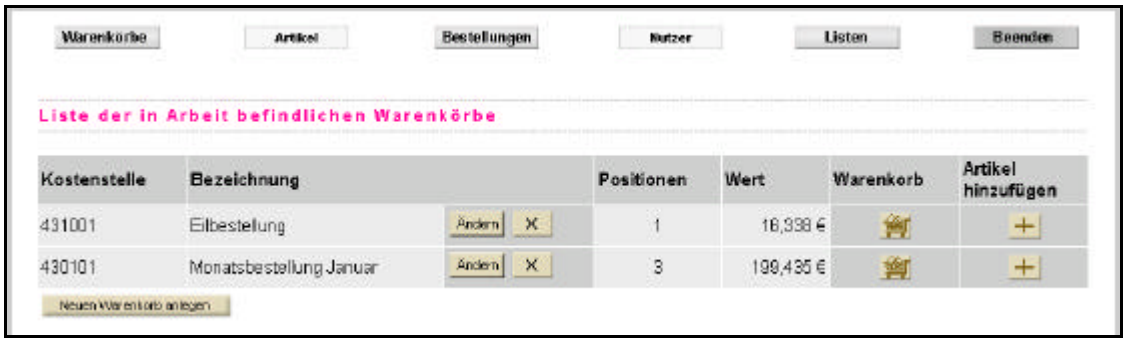


Abbildung 3-12: Screenshot der Warenkorb-Liste *(listeWarenkorb.jsp)*. Die Liste zeigt pro Warenkorb auch die Anzahl der in ihm liegenden Artikelpositionen und den Gesamtwert an.

Die Liste der Warenkörbe enthält sowohl Verzweigungsmöglichkeiten auf die unter ihm gespeicherten Positionen als auch in die Artikelsuche. Die Information, von welchem Warenkorb aus die Artikelsuche angesprungen wurde, wird in einer Session-Variablen gehalten und von der Anwendung genutzt, um zu erkennen, welchem Warenkorb sie einen ausgewählten Artikel hinzufügen muss. Wurde die Artikelsuche über die Menüführung unmittelbar angesprungen, wird der Nutzer bei Auswahl eines Artikels gebeten, einen der schon existierenden Warenkörbe zu wählen bzw. einen neuen anzulegen.

Abbildung 3-13: Screenshot der Bestellpositionen eines Warenkorbs (*warenkorbPositionen.jsp*). Der Nutzer erhält Informationen über den Preis pro Artikel und Position sowie die Preise pro Artikelgruppe sowie den Gesamtpreis der Bestellung.

Die Anzeige der Artikelpositionen eines Warenkorbs listet jede Position mit der Artikelnummer, Bestellmenge und Bezeichnung sowie dem Einzelpreis der jeweiligen Einheit und dem Gesamtpreis der Position auf. Die Summierung über Artikelgruppen und den Gesamtbestellwert gibt dem Nutzer weitere Informationen. Wie die Abbildung 3-13 zeigt, enthält die Liste auch Verzweigungsmöglichkeiten zur Eingabe der Lieferadresse, nach deren Eingabe die Bestellung ausgelöst werden kann. Weitere Verlinkungen führen zum Artikelkatalog und zur Eingabe von Artikeln, die im Katalog nicht zu finden sind. Das bei der Auswahl eines Artikels eingebbare Attribut *Bemerkung* wird aus Platzgründen in der Liste nicht angezeigt, ist jedoch durch das Anklicken des Buttons *Ändern* zu ersehen.

JSP	Erläuterung
ListeWarenkorb	*Zeigt eine Liste aller laufenden Warenkörbe an. Des weiteren sind hier die Verlinkungen zu den einzelnen Warenkorbfunktionen enthalten.*
listeWarenkorbUpdate	*gibt die Möglichkeit, den Warenkorb zu ändern und einen Warenkorb anzulegen.*
Lieferadresse	*ermöglicht das Eintragen einer Lieferadresse und das Auslösen der Bestellung*
warenkorbPositionen	*zeigt eine Liste aller Artikel des Warenkorbes an. Des weiteren sind Verlinkungen zur Eingabe der Lieferadresse, Artikelsuchfunktion und Eingabe unbekannter Artikel vorhanden.*
warenkorbPositionenUpdate	*ermöglicht, einen Artikel hinzuzufügen, zu entfernen oder zu ändern.*

Tabelle 2: Am Funktionsbereich *Warenkörbe* beteiligte JavaServer Pages

Von den JavaServer Pages genutzte Java Beans und Vererbungshierarchie der Java Beans

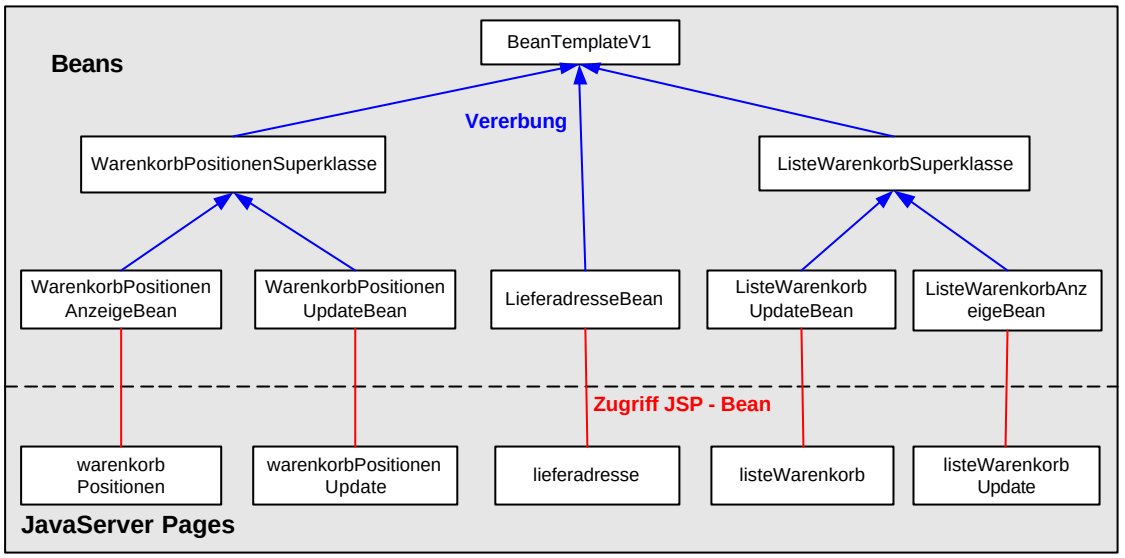


Abbildung 3-14: Klassenhierarchie und JSP - Java Bean Zuordnung des Funktionsbereichs *Warenkörbe*

Die beiden Superklassen *WarenkorbSuperklasse* und *ListeWarenkorbSuperklasse* implementieren eine Reihe von Variablen und Methoden, die von allen abgeleiteten Klassen benötigt werden. Dazu gehören bspw. Variablen, die aus der Datenbank gelieferte Werte zwischenspeichern, sowie die Methoden, die der erbenden Klasse einen Zugriff auf sie ermöglichen.

3.3.2.2 Funktionsbereich Artikelsuche

Der Menüpunkt Artikel / Artikelsuche[11] ermöglicht den Bestellern und Beschaffern, Artikel im Katalog zu suchen. Das Formular wird von der Datei *artikelSuchangaben.jsp* bereitgestellt. Der in das Feld Suchbegriff eingetragene Wert kann sowohl ein Teilstring einer Artikelbezeichnung sein als auch eine Artikelnummer. Logische Verknüpfungen verschiedener Strings sind aufgrund von fehlender Nachfrage nicht implementiert. Die Suche kann auf eine Kostenart oder Artikelgruppe eingeschränkt werden. Die Sortierung erfolgt entweder nach Bezeichnung des Artikels oder seiner Artikelnummer. Als weitere Option kann der Nutzer wählen, wie viele Artikel auf einer Antwortseite gelistet werden sollen. Möglich sind hier die Werte 20, 50, 100 und 1000. Werden mehr Treffer gefunden als auf einer Seite dargestellt werden können, erhält der Nutzer automatisch die Möglichkeit, in den Seiten des Suchergebnisses zu blättern. Die vorstehend beschriebenen Suchmöglichkeiten werden wie die Ergebnisse der Suche in einem eigenen Frame angezeigt.

[11] Nutzer der Gruppe Besteller sehen den Button Artikel, Nutzer des Sachgebiets 12 den Button Artikelsuche. Im Rahmen einer im ersten Quartal des Jahres 2003 durchzuführenden Vereinheitlichung der Begriffswahl der Beschaffungsanwendung werden die Begriffe zusammengeführt.

Abbildung 3-15: Trefferliste nach Eingabe des Suchworts *schilder*. Die Seite (*artikelSuchergebnis.jsp*) enthält mehr Optionen, wenn sie von einem Nutzer des Sachgebiets 12 aufgerufen wird.

Durch das Anklicken des Warenkorbsymbol in der Antwortseite des Suchergebnisses (*artikelSuchergebnis.jsp*) kann der Nutzer einen Artikel in den Warenkorb legen, nachdem er vorher gefragt wurde, in welcher Stückzahl er den Artikel möchte und die Gelegenheit erhält, der Bestellposition eine Bemerkung mitzugeben.

Im Gegensatz zu einem Nutzer des Sachgebiets 12 sehen Nutzer aus bestellenden Einrichtungen nur, ob ein Artikel vorrätig ist oder nicht. Das Sachgebiet 12 sieht hier die genaue Stückzahl der gelagerten Menge. Ihre Nutzer können über diese Liste auch die JavaServer Page *artikelUpdate* ansteuern, die es Ihnen ermöglicht Stammdaten eines Artikels zu ändern. Wie dem unten stehenden Screenshot des Änderungsdialogs zu entnehmen ist, sind die Manipulationsmöglichkeiten weitreichend. Insbesondere die Möglichkeit, Artikelmengen und –preise ohne Umweg über den Artikelzugang ändern zu können, ist eine historische Altlast und wird bei einem zukünftigen Release deaktiviert werden.

Abbildung 3-16: Änderungsdialog für die Stammdaten eines Artikels.

Wenn die Mitarbeiter des Sachgebiets 12 nicht möchten, dass ein Artikel von den Einrichtungen gesehen wird, bspw. weil er nicht mehr eingekauft werden kann, können sie ihn mit dem entsprechenden Flag aus der Anzeige der Einrichtungen entfernen.

JSP	Erläuterung
artikelSuchangaben	*ermöglicht dem Nutzer die Spezifikation von Suchparametern*
artikelSuchergebnis	*zeigt die Ergebnisse der Suche an und steuert den im Java Bean implementierten Blätter-Mechanismus. Sie enthält in Abhängigkeit der Nutzergruppe verschiedene weiterführende Links*
artikelUpdate	*ermöglicht das Anlegen, Bearbeiten und Deaktivieren von Artikeln*

Tabelle 3: Am Funktionsbereich *Artikel / Artikelsuche* beteiligte JavaServer Pages

Von den JavaServer Pages genutzte Java Beans und Vererbungshierarchie der Java Beans

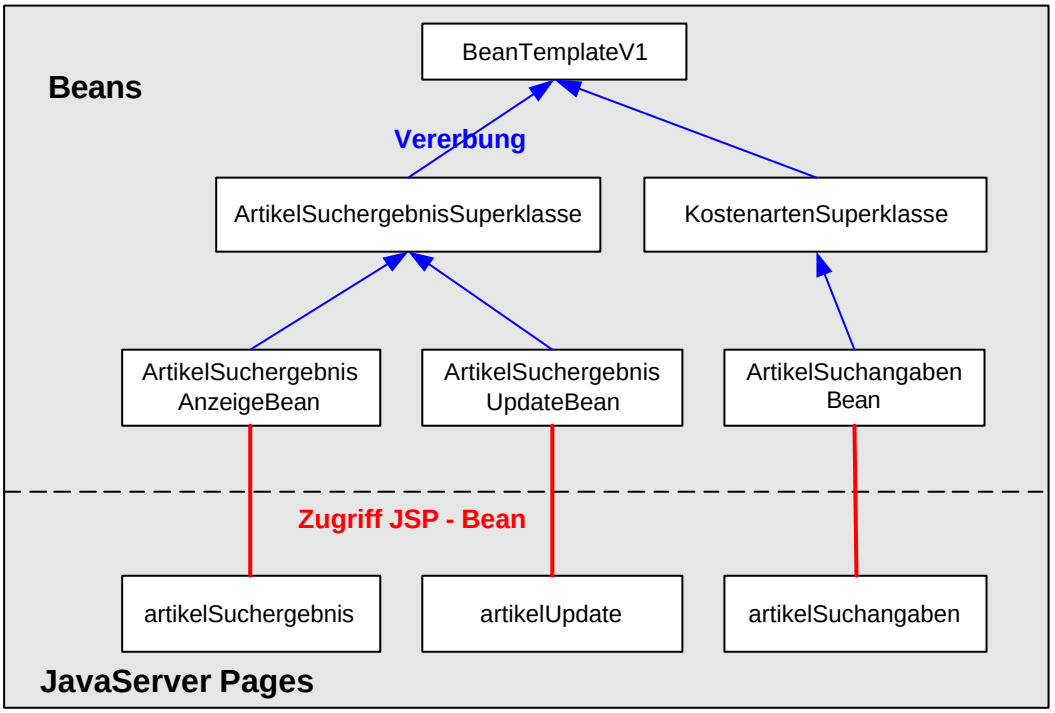


Abbildung 3-17: Klassenhierarchie und JSP - Java Bean Zuordnung des Funktionsbereichs *Artikel / Artikelsuche*

3.3.2.3 Funktionsbereich Bestellung

Innerhalb dieses Bereichs erhalten die Nutzer eine Übersicht der Bestellungen, die für die Kostenstellen, für die sie bestellberechtigt sind, ausgelöst wurden. Die Liste (*bestellungenListe.jsp*) enthält neben einigen Basisinformationen auch den Status der Bestellung (siehe Abschnitt 3.2.3 sowie Abbildung 3-7 zur Erläuterung der Statuswerte von Bestellungen in der Beschaffungsanwendung).

Die Abbildung 3-18 zeigt die Liste der neu eingegangenen Bestellungen. Weil der Nutzerzugang, mit dem der Screenshot durchgeführt wurde, der Nutzergruppe des Sachgebiets 12 angehört, werden die Bestellungen aller Kostenstellen angezeigt. Durch das Eingabefeld und die Select-Box im oberen Teil der Seite können ein Status und eine Kostenstelle gewählt werden, um die Liste einzuschränken.
Durch einen Klick auf den Button *Positionen* gelangt der Nutzer in das Bestellformular (*bestellungAnsicht.jsp*). Es ist im Aufbau bewusst dem papiernen Bestellformular (Abbildung 2-1) ähnlich, mit dem vor dem Einsatz der Anwendung gearbeitet wurde. Der Wiedererkennungsmoment wirkte sich bei der Einführung der Software positiv aus.

Abbildung 3-18: Liste der neu eingegangenen Bestellungen aus Sicht eines Nutzers des Sachgebiets 12.

Wie man in der Abbildung 3-19 sehen kann, enthält das Formular im oberen Teil einen Druck-Button, mit dem eine Bestellung ausgedruckt werden kann, sowie einen Navigationsbutton, der zurück zur Liste führt. Der Status der Bestellung kann entsprechend der Möglichkeiten, die weiter oben beschrieben wurden, verändert werden.
Im Kopf des eigentlichen Formulars finden sich einige konstante Informationen sowie die Kostenstelle und der Name bzw. die Lieferadresse des Bestellers. Sie wurden aus dem Nutzerstammdaten und den Eingaben in der JSP *lieferadresse.jsp* sowie dem Warenkorb generiert.
Die Mitarbeiter des Sachgebiets 12 haben durch das Anklicken des blau unterlegten Links der Mengenangabe die Möglichkeit, sowohl Bestellmenge als auch Bemerkung eines Artikels zu ändern (*bestellpositionUpdate.jsp*).

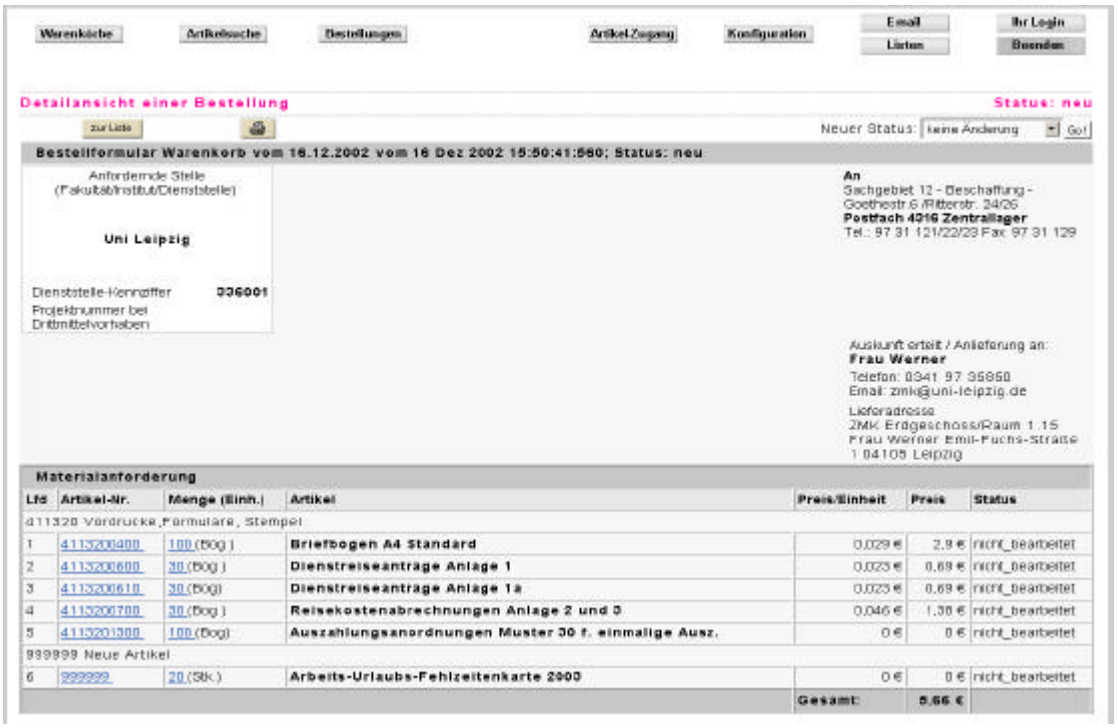


Abbildung 3-19: Darstellung einer Bestellung aus Sicht eines Nutzers des Sachgebiets 12 . Das Aussehen orientiert sich stark an dem der Internetseite zugrundeliegenden papiernen Bestellformular.

Sie können ebenfalls einen Artikel vollständig gegen einen anderen austauschen, wenn sie dem Link, der auf der Artikelnummer liegt, folgen. Sie gelangen dann automatisch in die bereits beschriebene Artikelsuche. Die Anwendung merkt sich, dass die Artikelsuche mit dem Ziel aufgerufen wurde, einen bestellten Artikel gegen einen anderen auszutauschen und bietet zu diesem Zweck neben jedem gelisteten Artikel einen speziellen Button an, der bei dem üblichen Aufruf der Artikelliste nicht zu sehen ist. Jede Änderung der ursprünglich bestellten Artikelposition wird im Bestellformular kenntlich gemacht. Um einen Änderungsvorgang transparent zu machen, werden sowohl die ursprüngliche Position als auch die geänderte zusammen angezeigt.
Entsprechend dem Warenkorb enthält auch das Bestellformular Summen pro Artikelgruppe und eine Gesamtsumme.

JSP	Erläuterung
bestellungenListe	*zeigt eine Liste von Bestellungen an*
bestellungAnsicht	*zeigt die Positionen der gewählten Bestellung und bietet die Möglichkeit den Status der Bestellung zu ändern, sowie Bestellpositionen zu verändern (Nutzergruppe Sachgebiet 12)*
bestellpositionUpdate	*bietet das Änderungsformular der Bestellpositionen an*

Tabelle 4: Am Funktionsbereich *Bestellung* beteiligte JavaServer Pages

Von den JavaServer Pages genutzte Java Beans und Vererbungshierarchie der Java Beans

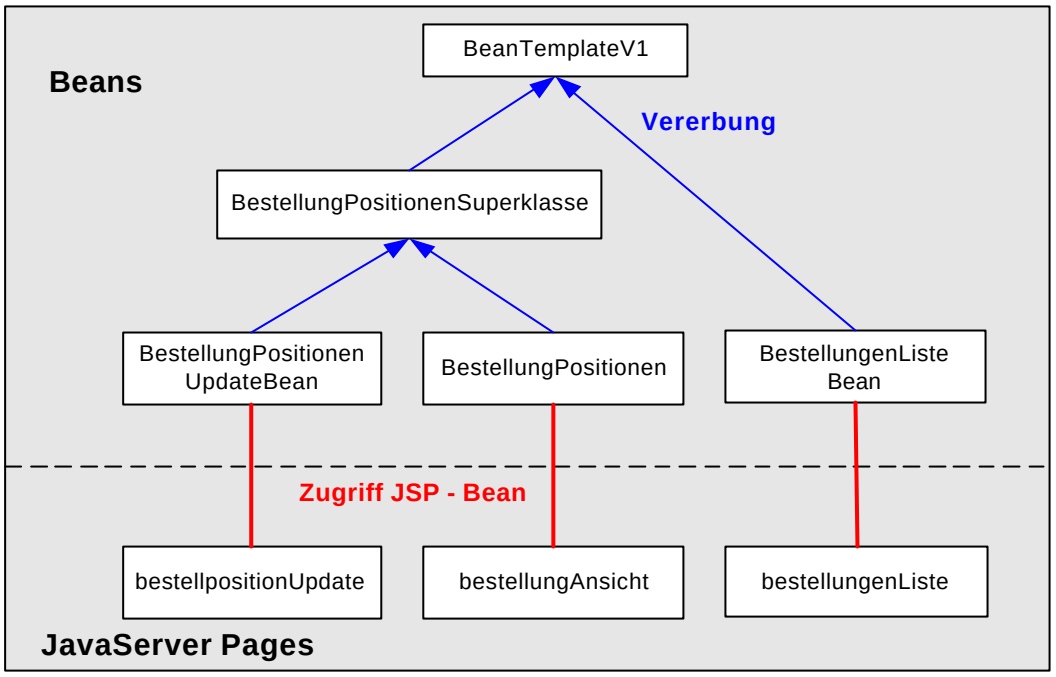


Abbildung 3-20: Klassenhierarchie und JSP - Java Bean Zuordnung des Funktionsbereichs *Bestellungen*

Der Einsatz einer Superklasse für das Bean: BestellungenListeBean schien nicht sinnvoll, da kein weiteres Bean geplant war, an die eine Vererbung sinnvoll wäre. Die entsprechenden Variablen und Methoden sind deshalb Bestandteil des Beans.

3.3.2.4 Funktionsbereich Guthaben

Die *Guthaben*-Funktion ermöglicht den Nutzern des Sachgebiets 12, Kostenstellen ein Guthaben pro Kostenart zuzuordnen. Dies wird den Bestellsummen in den Auswertungslisten für die jeweiligen Kostenstellen und Kostenarten gegengerechnet und ermöglicht dem für die Bestellungen verantwortlichen Nutzer eine Kontrolle seiner Ausgaben.

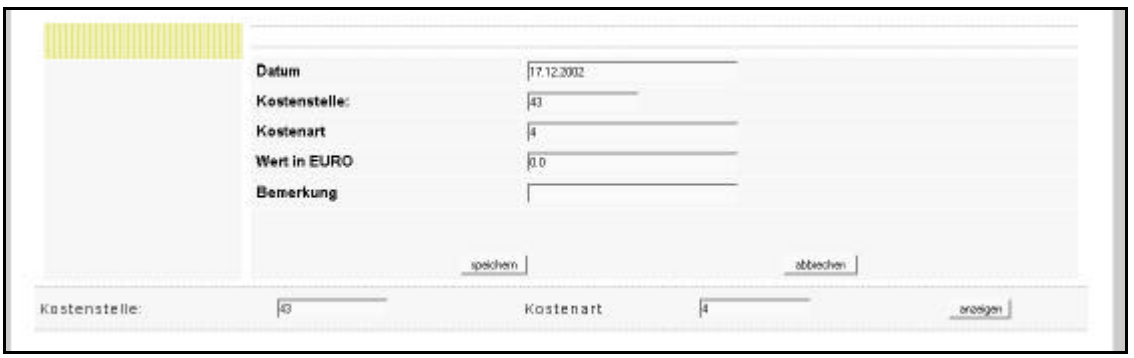


Abbildung 3-21: Eingabemaske der Guthaben. Die Maske kombiniert sowohl die Eingabemöglichkeit als auch die Listung von Guthaben.

Guthaben entstehen durch Überhänge von Budgets aus den vorangegangenen Jahren. Die Einrichtungen bitten das Sachgebiet 12 in Einzelfällen, bereits Umbuchungen von Geldern vorzunehmen, bevor Artikel für dieses Geld bestellt wurden. Diese Überhänge werden durch die Guthaben-Funktion abgebildet.

JSP	Erläuterung
GuthabenEingabe	*bietet die Möglichkeit, ein Guthabe, für eine Kostenart und Kostenstelle einzugeben. Des weiteren die Möglichkeit, sich alle vorhandenen Guthaben für eine Relation (Kostenstelle/Kostenart) auszugeben*

Tabelle 5: Am Funktionsbereich *Bestellung* beteiligte JavaServer Pages

Von den JavaServer Pages genutzte Java Beans und Vererbungshierarchie der Java Beans

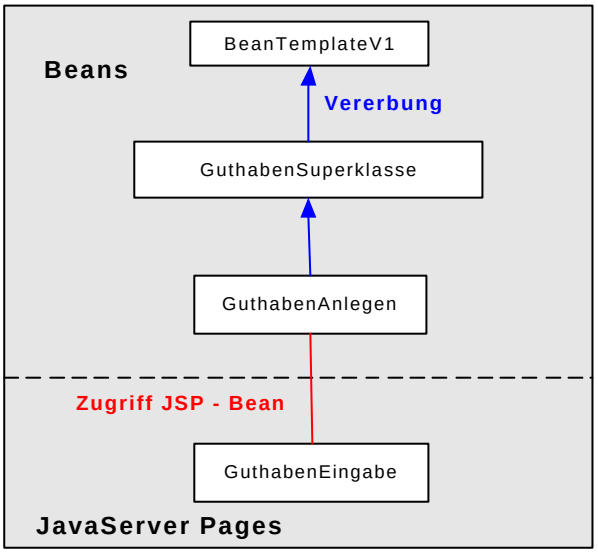


Abbildung 3-22: Klassenhierarchie und JSP - Java Bean Zuordnung des Funktionsbereichs *Guthaben*

3.3.2.5 Funktionsbereich Artikel-Zugang

Der Artikel-Zugang ermöglicht den Mitarbeitern des Sachgebiets 12 die Eingabe von Lieferungen. Bei der Eingabe werden die gelieferten Positionen mit dem Artikelstamm verknüpft. Die Höhe des Artikelpreises und des Lagerbestands werden automatisch korrigiert. Während der Lagerbestand um die gelieferte Menge ergänzt wird, übernimmt das Beschaffungssystem den Preis aus der letzten den Artikel betreffenden Lieferung.

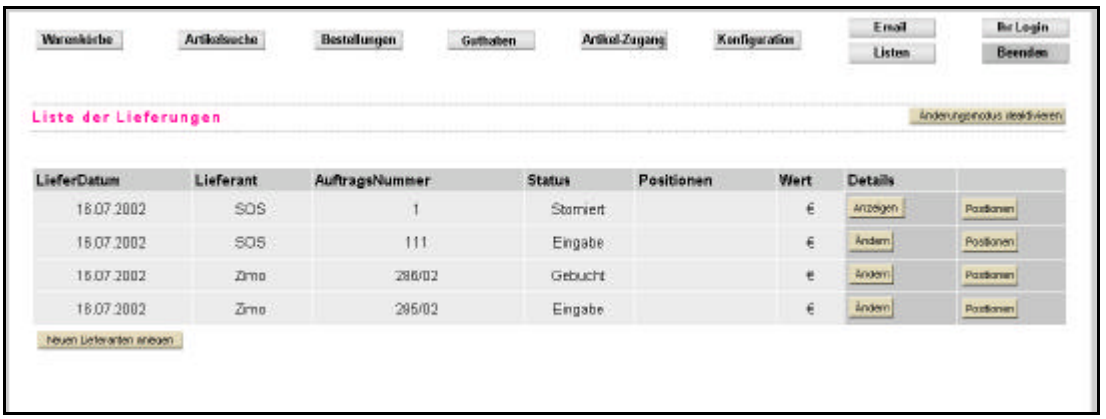


Abbildung 3-23: Screenshot der Lieferungsliste (*LieferungenListe.jsp*, in Entwicklung). Neben dem Lieferanten und der Auftragsnummer enthält die Liste auch Informationen zum Status, der Anzahl der gelieferten Positionen und ihrem Wert (in der Implementierungsphase).

Das ursprüngliche Vorgehen vor Einsatz der Beschaffungsanwendung ermittelte jeweils einen Durchschnittswert (siehe Abschnitt 2.3.2, Erklärung zu Bild 6). Aus Gründen der höheren Nachvollziehbarkeit seitens des Sachgebiets 12 wurde das Verfahren wie oben erläutert geändert.
Jede Lieferung hat drei mögliche Stati: *Eingabe*, *Gebucht* und *Storniert*. Der Status *Eingabe* ist der Default-Wert einer Lieferung, nachdem sie angelegt wurde. Sobald alle Lieferpositionen eingegeben wurden, führt das Umsetzen des Statuswerts auf *Gebucht* zu den oben genannten Aktualisierungen des Artikelstamms. Der Status *Storniert* macht diese Änderungen wieder rückgängig.
Die Nutzer des Sachgebiets 12 haben die Wahl zwischen zwei Anzeigvarianten der Dialoge des Funktionsbereichs. Erst wenn sie den Änderungsmodus aktiviert haben, erhalten sie schreibenden Zugriff auf die Daten. Analog zu anderen Bereichen der Beschaffungsanwendung, die einen ähnlichen Mechanismus einsetzen, soll die Einrichtung dieses Zwischenschritts gegen ungewollte Veränderungen von Werten schützen.
Durch das Anklicken des Buttons *Ändern* können die Stammdaten einer Lieferung geändert werden (Lieferant, Rechnungsnummer, etc.). Der Button Positionen ermöglicht das Verzweigen in die Liste der Lieferpositionen (Abbildung 3-24). In die Liste der Lieferpositionen könne nur solche Artikel aufgenommen werden, die bereits im Artikelstamm enthalten sind. Neue Artikel müssen dem Artikelstamm erst über den *Funktionsbereich Artikelsuche* (Abschnitt 3.3.2.2) zugefügt werden.

Dieser Funktionsbereich befindet sich in der Entwicklungsphase und soll ab Februar 2003 zum Einsatz kommen.

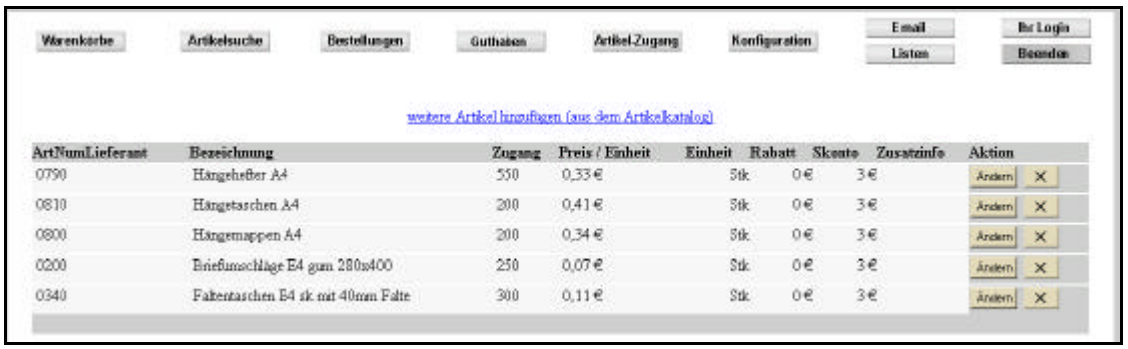


Abbildung 3-24: Screenshot der Liste der Lieferungspositionen. Die Liste befindet sich kurz vor dem Abschluss des Entwicklungsstadiums (*LieferungPositionen.jsp*).

JSP	Erläuterung
LieferungenListe	*zeigt die Liste aller Lieferungen aus*
LieferungenUpdate	*ermöglicht, die Stammdaten eine Lieferung zu ändern oder eine neue Lieferung anzulegen*
LieferungPositionen	*gibt eine Liste aller Positionen einer ausgewählten Lieferung zurück*
LieferungPositionenUpdate	*ermöglicht, neue Artikel in eine Lieferung einzutragen und bei bestehenden Lieferpositionen Änderungen vorzunehmen. Des weiteren kann hier der Status der Lieferung geändert werden*

Tabelle 6: Am Funktionsbereich *Artikel-Zugang* beteiligte JavaServer Pages

Von den JavaServer Pages genutzte Java Beans und Vererbungshierarchie der Java Beans

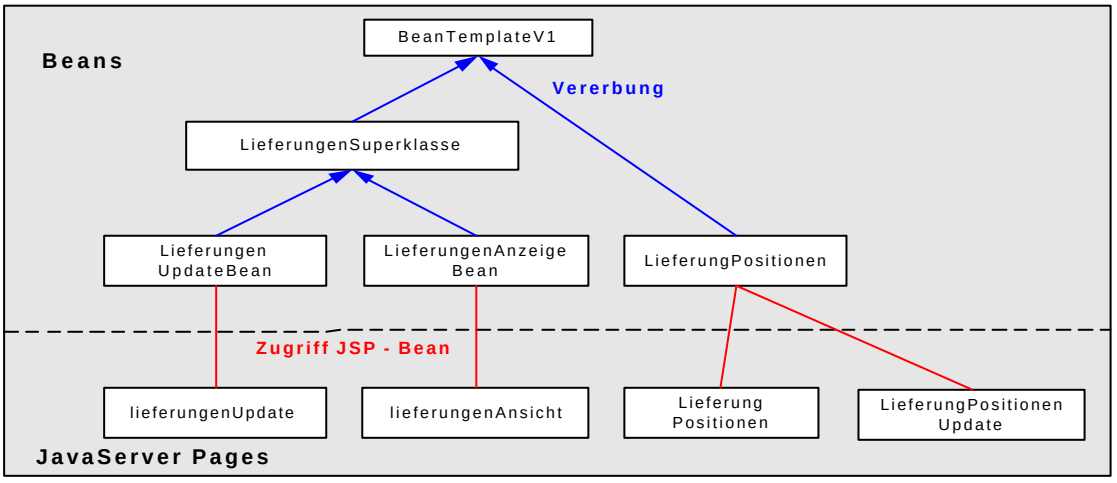


Abbildung 3-25: Klassenhierarchie und JSP - Java Bean Zuordnung des Funktionsbereichs Lieferungen (Entwicklungsphase)

3.3.2.6 Funktionsbe reich Konfiguration

Der Funktionsbereich Konfiguration subsummiert einige Einstellungen, die zum Betrieb des Anwendung wichtig sind, aber eine geringere Nutzungsfrequenz haben als bspw. die Bereiche Artikelsuche oder Bestellungen. Im Konfigurationsbereich können Kostenartenzuodnungen und -bezeichnungen geändert werden, sowie neue Kostenarten angelegt werden. Die gleichen Möglichkeiten werden für die Kostenstellen angeboten.

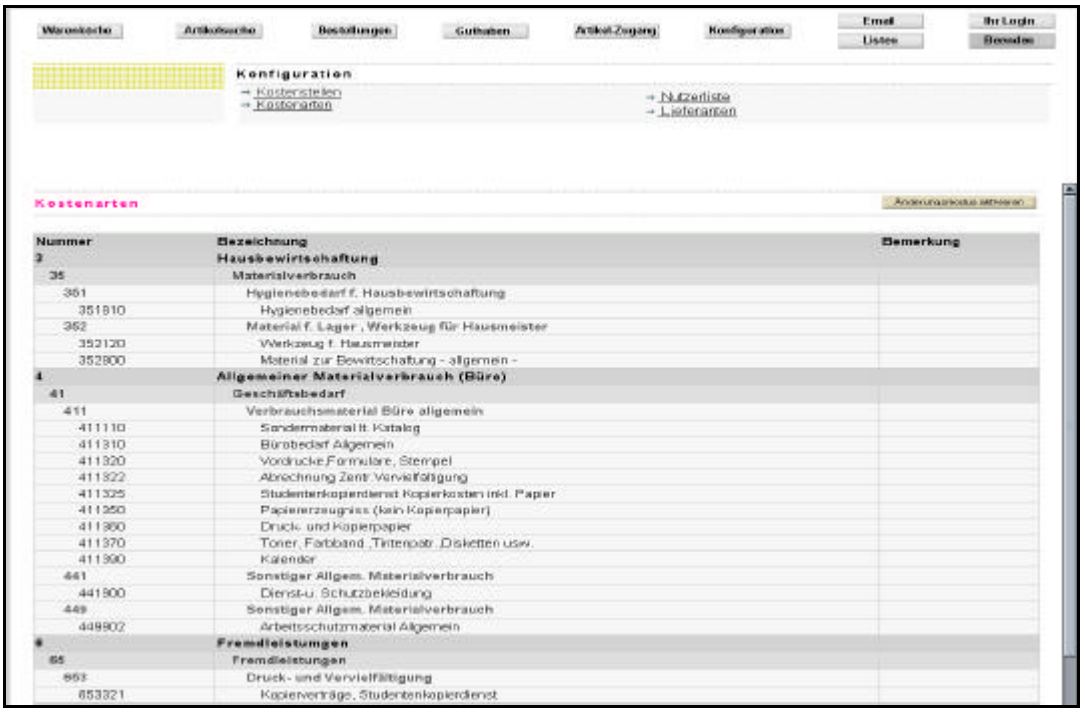


Abbildung 3-26: Screenshot des Funktionsbereichs Konfiguration. Der untere Frame zeigt die Liste der Kostenarten an. Nach dem Aktivieren des Änderungsmodus (Klick auf gleichlautenden Button) gelangen die Sachbearbeiter des Sachgebiets 12 in die Änderungsmasken.

Die Nutzerliste ermöglicht dem Sachgebiets 12, neue Nutzer anzulegen, Stammdaten zu ändern, neue Passwörter zu vergeben und die Kostenstellen einzurichten, für die ein Nutzer bestellen darf.
Der Untermenüpunkt Lieferanten verwaltet den Lieferantenstamm. Er wird benötigt, um Lieferungen eingeben zu können.

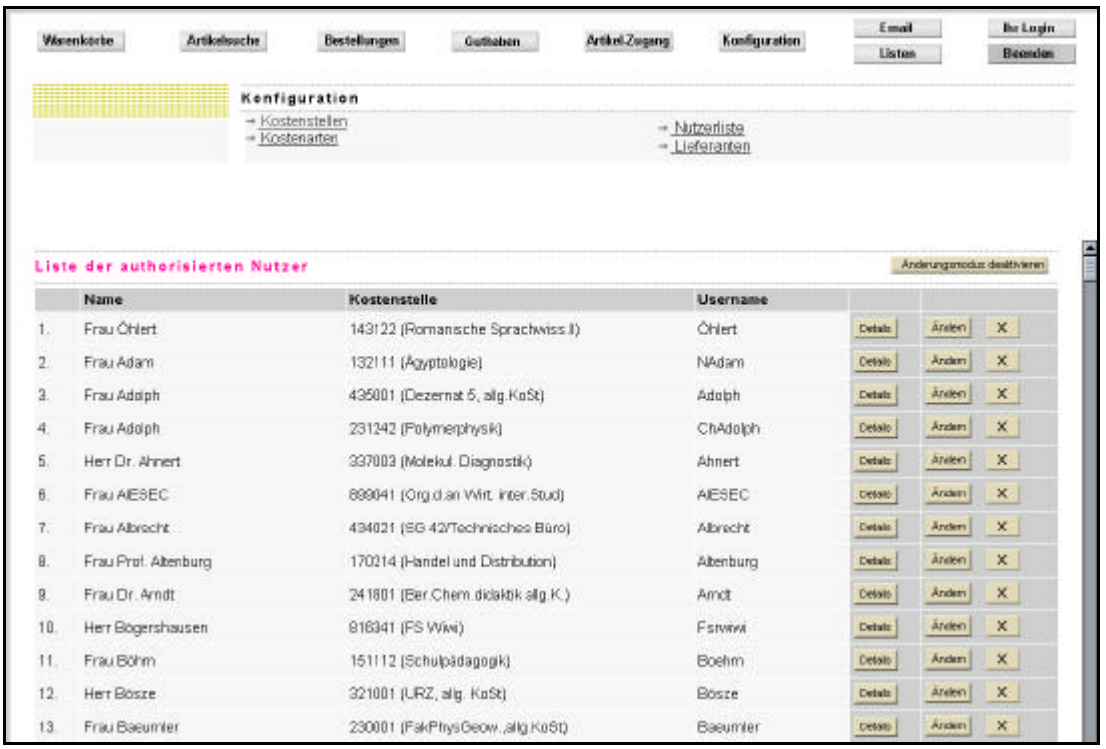


Abbildung 3-27: Screenshot der Nutzerliste in aktiviertem Änderungsmodus. Der Button *Ändern* führt zu einer Maske zur Änderung der Nutzerstammdaten. Hinter dem Button Details versteckt sich der Eingabedialog zur Verwaltung der Kostenstellen, für die ein Nutzer bestellberechtigt ist.

JSP	Erläuterung
kostenstelle	*zeigt alle Kostenstellen an und bietet die Möglichkeit, sie zu ändern oder neue anzulegen*
kostenarten	*zeigt alle Kostenarten an*
kostenarten_Update	*bietet die Möglichkeit, Kostenarten zu ändern, anzulegen und die Hierarchie der Kostenarten zu ändern*
user	*zeigt alle Nutzer in einer Liste an*
userDetails	*gibt die Stammdaten eines ausgewählten Nutzers an und erlaubt die Eingabe der Kostenstellen, für die er bestellberechtigt ist*
userUpdate	*ist zuständig für das Ändern, Löschen und Anlegen eines Nutzers*
lieferanten	*gibt eine Liste aller Lieferanten aus*
lieferantenUpdate	*ist zuständig für das Ändern, Löschen und Anlegen eines Lieferanten*

Tabelle 7: Am Funktionsbereich *Konfiguration* beteiligte JavaServer Pages

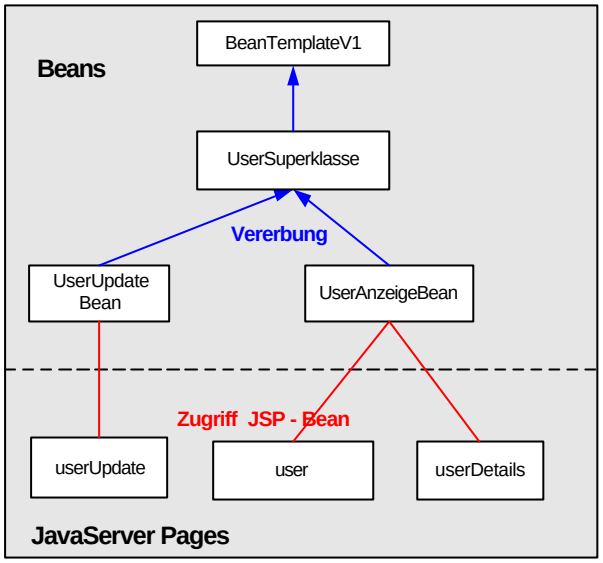


Abbildung 3-28: Klassenhierarchie und JSP - Java Bean. Zuordnung des Unterbereichs *Nutzerlister*

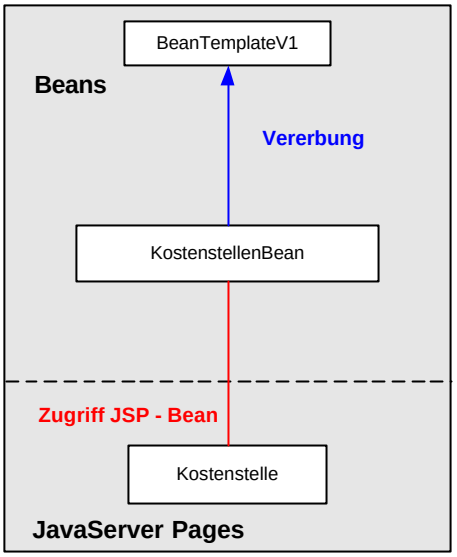


Abbildung 3-29: Klassenhierarchie und JSP - Java Bean Zuordnung des Unterbereichs *Kostenstellen*

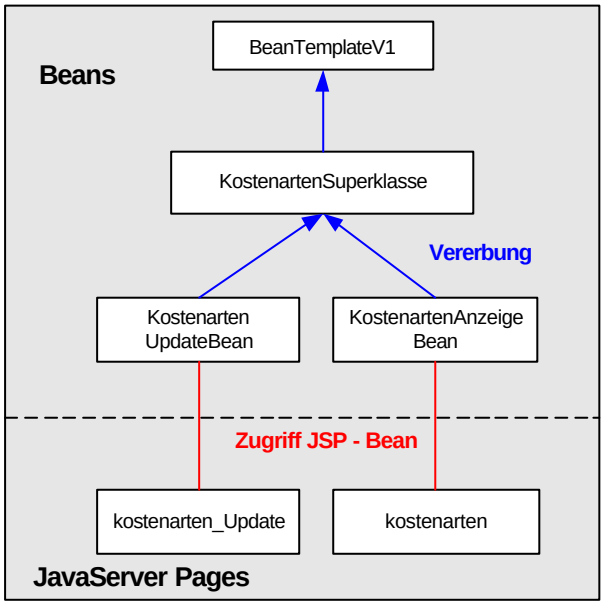


Abbildung 3-30: Klassenhierarchie und JSP - Java Bean Zuordnung des Unterbereichs *Kostenarten*

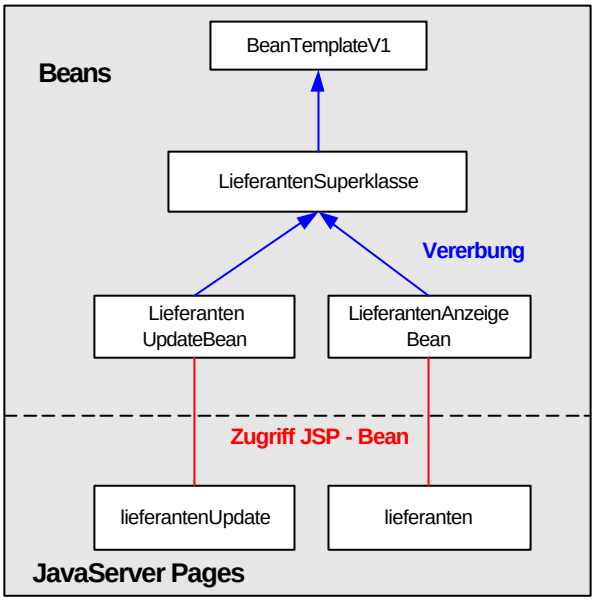


Abbildung 3-31: Klassenhierarchie und JSP - Java Bean Zuordnung des Unterbereichs *Lieferanten*

3.3.2.7 Funktionsbereich Email

In dem Funktionsbereich Email haben die Nutzer des Sachgebiets 12 die Möglichkeit, Emails an alle oder einzelne, bzw. Gruppen von Nutzern der Beschaffungsanwendung zu verschicken. Sie haben ebenfalls die Möglichkeit, Dateien an die Mails anzuhängen und sich in der Vergangenheit versendete Mails anzusehen.

Abbildung 3-32: Das Email-Formular ermöglicht die Eingabe und das Versende von Emails an alle oder einzelne Nutzer der Beschaffungsanwendung.

JSP	Erläuterung
EmailHistory	*zeigt eine Liste aller Emailvorgänge die der User getätigt hat und bietet die Möglichkeit,sie zu wiederholen*
Email	*ermöglicht die Eingabe einer Email, die Auswahl der Adressaten, die Auswahl des Anhangs und das Versenden der Nachricht*
UploadAttachement	*überträgt eine Datei auf den Server, um sie an einen Mailtext anhängen zu können*

Tabelle 8: Am Funktionsbereich *Email* beteiligte JavaServer Pages

Von den JavaServer Pages genutzte Java Beans und Vererbungshierarchie der Java Beans

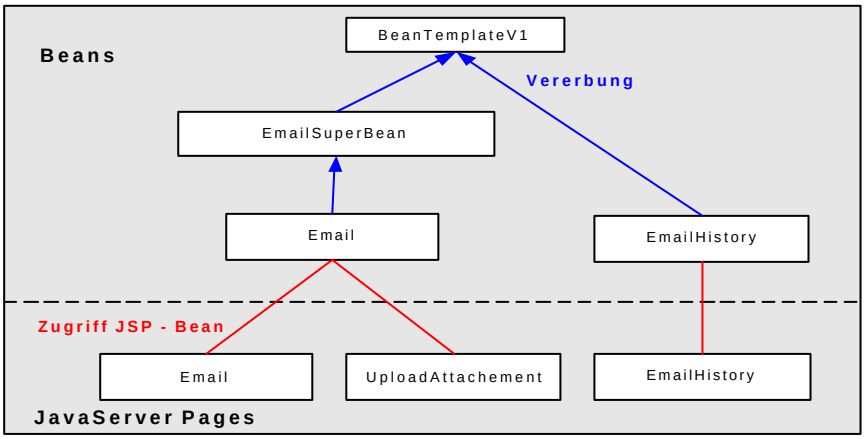


Abbildung 3-33: Klassenhierarchie und JSP - Java Bean Zuordnung des Funktionsbereichs *Email*

3.3.2.8 Funktionsbereiche Mein Login

In diesem Bereich können Nutzer Ihre Stammdaten einsehen und sich einen neuen Login-Namen sowie ein neues Passwort geben.

Abbildung 3-34: Screenshot der Stammdaten eines Nutzers mit der Option, Login-Namen und Passwort zu ändern. Neben den Stammdaten werden auch die Kostenstellen angezeigt, für die der Nutzer bestellberechtigt ist.

JSP	Erläuterung
userDaten	*zeigt die Stammdaten des eingeloggten Nutzers sowie die Kostenstellen, für die er bestellberchtigt ist, an und bietet ihm die Möglichkeit, Loginnamen und Passwort zu ändern*

Tabelle 9: Am Funktionsbereich *Mein Login* beteiligte JavaServer Pages

Von den JavaServer Pages genutzte Java Beans und Vererbungshierarchie der Java Beans

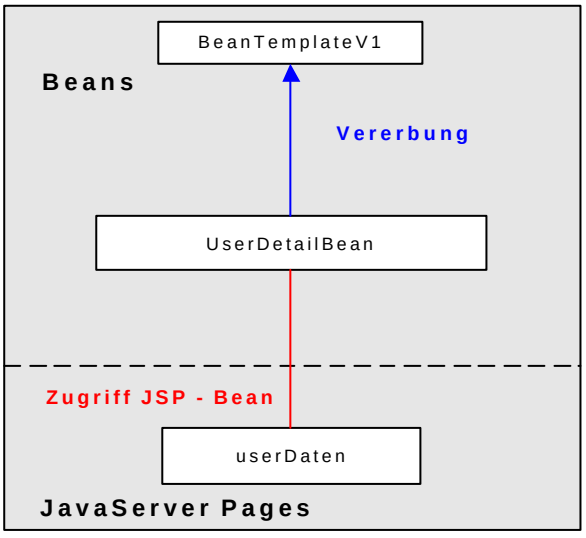


Abbildung 3-35: Klassenhierarchie und JSP - Java Bean Zuordnung des Funktionsbereichs *Ihr Login*

3.3.2.9 Funktionsbereich Listen

In diesem Bereich können Nutzer verschiedene Auswertungen generieren und anzeigen lassen. Grundsätzlich ist es bei allen kostenstellenbezogenen Listen so, dass ein Nutzer der Einrichtungen nur diejenigen Informationen einsehen kann, die Kostenstellen betreffen, für die er bestellberechtigt ist. Das Sachgebiet 12 kann allerdings Nutzern das Recht einräumen, Auswertungen über eine gesamte Fakultät zu generieren. Die Notwendigkeit, diese Möglichkeit zu implementieren ergab sich, weil einzelne Nutzer neben den Bestellungen für bestimmte Kostenstellen auch mit der Aufgabe betraut sind, das Gesamtbestellvolumen einer Fakultät zu überwachen.

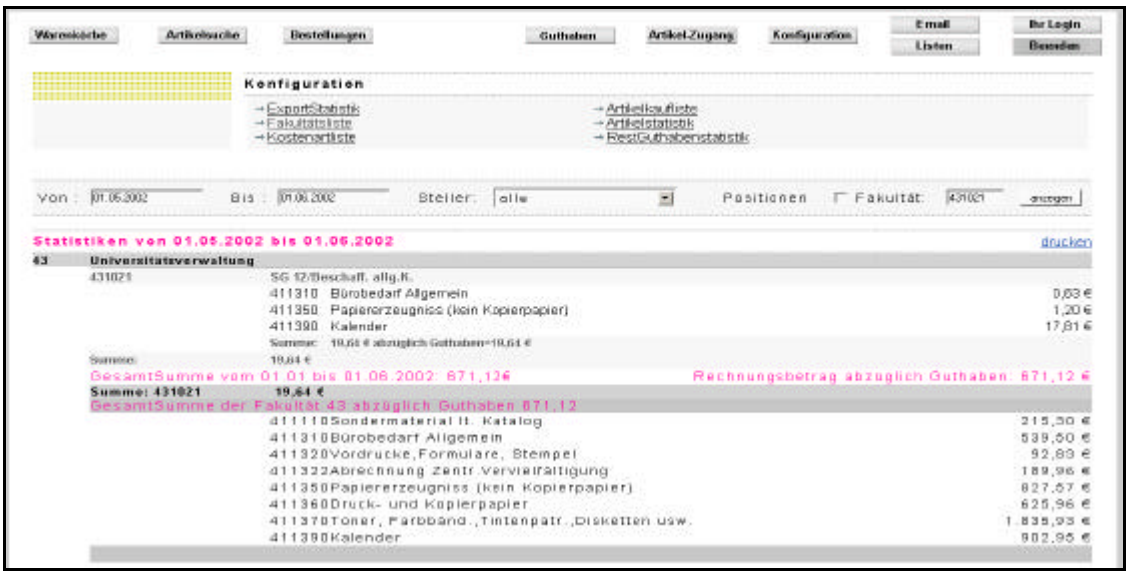


Abbildung 3-36: Screenshot der Auswertung *Fakultätsliste* für die Kostenstelle 431021 für den Monat Mai 2002. Das Feld Steller ermöglicht die Auswahl von sechstelligen (Institute, Fakultäten, Dezernate), sieben- oder achtstelligen Kostenstellen (Drittmittelprojekte). Wird das Feld Positionen angeklickt, werden die Artikel-Positionen der Artikelgruppen ebenfalls ausgegeben. Da der eingeloggte Nutzer zum Sachgebiet 12 gehört, kann er ebenfalls die Summen der Fakultät über Bestellungen in alle Kostenarten innerhalb des genannten Zeitraum sehen.

JSP	Erläuterung
statistik_fuer_export	*Zeigt die Liste der Buchungswerte aller buchbereiten Bestellungen. (Link: ExportStatistik)*
statistik_von_export	*Zeigt eine Liste aller Listen, die exportiert wurden. (Link: ExportStatistik)*
statistik.jsp	*subsummiert Bestellungen für einen wählbaren Zeitraum unter ihrer Kostenstelle und Kostenart. (Link: Fakultätsliste)*
Koartstatistik	*zeigt eine Liste aller Kostenarten, für die innerhalb eines definierten Zeitraum bestellt wurde. (Link: Kostenartliste)*
Artikelkaufstatistik	*Diese jsp zeigt an, wie oft ein einzelner Artikel in einem Zeitraum bestellt worden ist. (Link: Artikelkaufliste)*
Artikelstatistik	*zeigt an, wie oft ein einzelner Artikel aus einem vorher definierten Kostenart in einem Zeitraum bestellt worden ist. (Link: Artikelstatistik)*
Restguthabenstatistik	*zeigt die Guthaben von Kostenstellen / Kostenarten abzüglich der bereits ausgelösten Bestellungen in einem Zeitraum*

Tabelle 10: Am Funktionsbereich *Listen* beteiligte JavaServer Pages

Von den JavaServer Pages genutzte Java Beans und Vererbungshierarchie der Java Beans

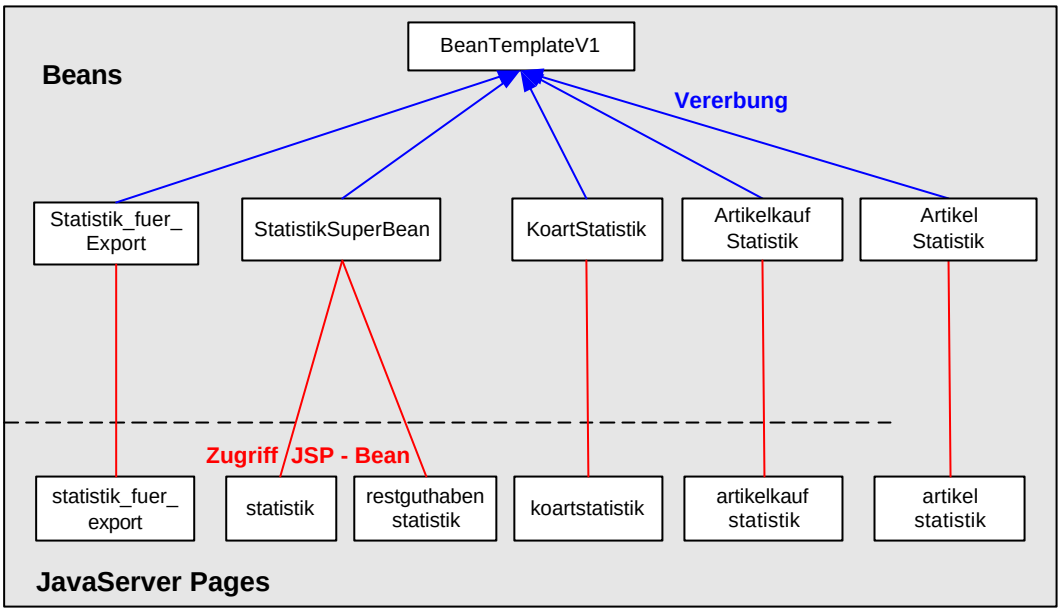


Abbildung 3-37: Klassenhierarchie und JSP - Java Bean Zuordnung des Funktionsbereichs *Listen*

3.3.3 Datenübernahme aus der Legacy-Anwendung HIS BEL

Im Abschnitt 2.3 wird die Lagerverwaltungssoftware HIS BEL vorgestellt. Aus Sicht der neu entwickelten Beschaffungslösung war es interessant, die in HIS BEL verwalteten Artikel- und Bestandsdaten zu übernehmen. Da es sich bei der dem Programm um eine ursprünglich auf dem dBase-Standard basierende Anwendung handelt, konnten die Daten problemlos über einen entsprechenden ODBC-Treiber, der in dem Betriebssystem Windows 2000 mitgeliefert wird ausgelesen werden. Die Tabelle: ANS.DBF beinhaltete alle Artikeldaten, die sich zur Übernahme in die Beschaffungsanwendung anboten.

3.3.4 Erreichbarkeit und Betrieb der Beschaffungsanwendung

Die Beschaffungsanwendung ist über Verlinkungen auf den Internetseiten der Universität im Bereich der Internetseiten des Dezernats 1 zu erreichen.
Die Adresse lautet:

http://procurement.ag-server.de/uni-leipzig/index.jsp

Die Applikation läuft im Produktionsbetrieb auf einem kombinierten Web- und Application-Server. Als Webserver wird der auf dem Apache-Webserver, Version 1.3.19 basierende IBM-Webserver mit gleicher Versionsnummer eingesetzt. Als Application-Server ist Tomcat, Version 3.3.a im Einsatz.

Bei der Hardware handelt es sich um einen Xeon-Dualprozessor mit 1 GHz. Als Datenbankserver ist ein Pentium 3 mit 1,2 GHz im Einsatz. Die Konfiguration wird durch eine identische Fallbacklösung abgesichert.

4 Vergleich ausgewählter Application Server

Die Beschaffungsanwendung wurde auf Basis des Application-Servers Apache Tomcat, Version 3.3 entwickelt. Er sollte als Referenzimplementierung von Servlet- und JavaServer Page – Spezifikationen eine gute *Ausgangsposition für den Einsatz der Anwendung* auch in der Laufzeitumgebung anderer Application-Server sein. Tatsächlich zeigte sich, dass es nach einer kurzen Einarbeitungszeit in die nachstehend genannten Administrationstools möglich war, die Beschaffungsanwendung in der Laufzeitumgebung des *Websphere Application Server, Version 3.5 von IBM* auszuführen. Es waren keinerlei Anpassungen der Anwendung notwendig.
Die Motivation für die Portierung der Anwendung entspringt der Frage, wie in technischer Hinsicht das *Verhältnis* des open-source - Produkts Apache Tomcat zu dem kostenintensiven Application-Server von IBM, als einem der Weltmarktführer in diesem Produktsegment einzuschätzen ist. Die Bandbreite der vergleichbaren Eigenschaften von Application-Servern ist gross. Das Antwortverhalten unter verschiedenen Lastszenarien, Stabilitätseigenschaften, Transaktionskonzepte und Skalierbarkeit sowie die Administrationsmöglichkeiten sind Merkmale, die viele der über 30 Anbieter von Application-Servern ihren Produkten zuordnen [FL02].
Da eine Untersuchung aller Eigenschaften von Application-Servern den Rahmen der vorliegenden Arbeit sprengen würde, geht dieses Kapitel ausschließlich auf die Bereiche des *Antwortverhaltens unter Lastbedingungen* und *der Administrationsmöglichkeiten* näher ein. Der letztgenannte Punkt gibt durch eine umfassende Beschreibung der Administrationstools auch einen kleinen Einblick in das Einsatzgebiet des IBM-Produkts. Die Untersuchungen zum Antwortverhalten der Application-Server unter Lastbedingungen fokussieren das Session-Management als eine wesentliche Eigenschaft der Funktionalität von Application-Servern unter Performance-Gesichtspunkten (siehe auch Stichwort *Session-Management* im Anhang, Abschnitt B).
Den Untersuchungen ist eine kurze Vorstellung des in der vorliegenden Arbeit bisher kaum behandelten Application-Servers von IBM vorweggestellt..

4.1 IBM Websphere vs. Apache Tomcat

Websphere stellt eine von der Firma IBM produzierte Produktfamilie dar, mit der webbasierte Applikationen entwickelt, getestet und ausgeführt werden können. Sie besteht aus insgesamt drei Bereichen:

- Entwicklungsumgebungen, bspw. Websphere Studio und VisualAge for Java
- Laufzeitumgebung mit dem Hauptbestandteil des Websphere Application Server
- Verwaltungs- und Optimierungssoftware, bspw. das Performance Pack

Der Kern bildet dabei der javabasierte Application Server. Er entstand aus *Servlet Express*, einer Java-Laufzeitumgebung für Servlets, die kostenlos zur Verfügung gestellt wurde. Nachdem dieses Produkt eine positive Resonanz fand, wurde es zum Websphere Application Server erweitert. [ES01, Seite 3, 4]
Der Application Server der Version 3.5 wird in drei Ausbaustufen für unterschiedliche Bedürfnisse und Skalierungen angeboten:

Standard Edition
Die Standard Edition, auch Standard Application Server genannt, besitzt einen Web-Container, mit der die Web-Komponenten Java Servlets und JavaServer Pages unterstützt werden. Weiterhin beherrscht die Standard Edition auch XML und Session-Management. Die Standard Edition ist auf eine Ein-Maschinen-Umgebung für Servlets und JavaServer Pages beschränkt, bietet aber den Zugriff von mehreren Clients.

Advanced Edition
Die Advanced Edition, welche auch als Advanced Application Server bezeichnet wird, enthält die gesamte Funktionalität der Standard Edition, besitzt aber zusätzlich einen EJB-Container zum Einsatz von Enterprise JavaBeans und kann mit einem Pool von JVMs arbeiten.

Enterprise Edition
Als Basis für die Enterprise Edition dient die Advanced Edition. Zusätzlich beinhaltet diese Version das Programm *Component Broker*, welches eine Implementierung des CORBA Standards ist, der von der Object Management Group (OMG) stammt. Desweiteren wurde im Component Broker die EJB-Spezifikation gesondert implementiert.

Die Positionierung von Tomcat wurde bereits im Rahmen der Entwicklung der Beschaffungsanwendung in Abschnitt 3.1.2 erläutert. Tomcat selbst ist Bestandteil des Jakarta-Projekts der Apache Software Foundation. Innerhalb dieses Projekts gibt es viele Unterprojekte, die mit den von der Websphere-Familie angebotenen Funktionalitäten vergleichbar sind. [JT02]
Für die folgenden Vergleichsszenarien wurde jedoch ausschließlich der JSP/Servlet – Container Tomcat 3.3 eingesetzt.

4.2 Komponenten und Administrationsmodell

Die im Verlauf der Arbeit bereits erläuterte Architektur von Application-Servern bringt bei ihrer Abbildung auf eine Produktionsumgebung administrativen Aufwand mit sich. Durch die Skalierung

einer Anwendung bspw. durch den per Loadbalancing gesteuerten Einsatz mehrerer Application-Server auf verschiedenen Rechnersystemen kann der Aufwand so groß werden, dass die Möglichkeit einer zentralen Verwaltung der Application-Server ein wichtiges Kriterium für die Wahl eines Produkts ist.

4.2.1 Websphere Komponenten

Im Websphere Application Server wird die Verwaltung von einer Gruppe von Host-Maschinen, welche *verwaltete Knoten* genannt werden, durchgeführt. Jeder verwaltete Knoten führt einen Verwaltungs- bzw. Administrationsserver innerhalb einer eigenen JVM aus. Ein Verwaltungsserver hat das Überwachen, Konfigurieren und Verwalten, insbesondere die Speicherverwaltung, von Ressourcen an diesem Knoten zur Aufgabe. Zu diesen Ressourcen zählen Objekte wie Web- und EJB-Container, Servlets, JSP-Dateien, JavaBeans und Enterprise JavaBeans. Permanente Daten, die zu einer Ressource gehören, werden in einem zentralen Daten-Repository gespeichert. [ES01, Seite 14].

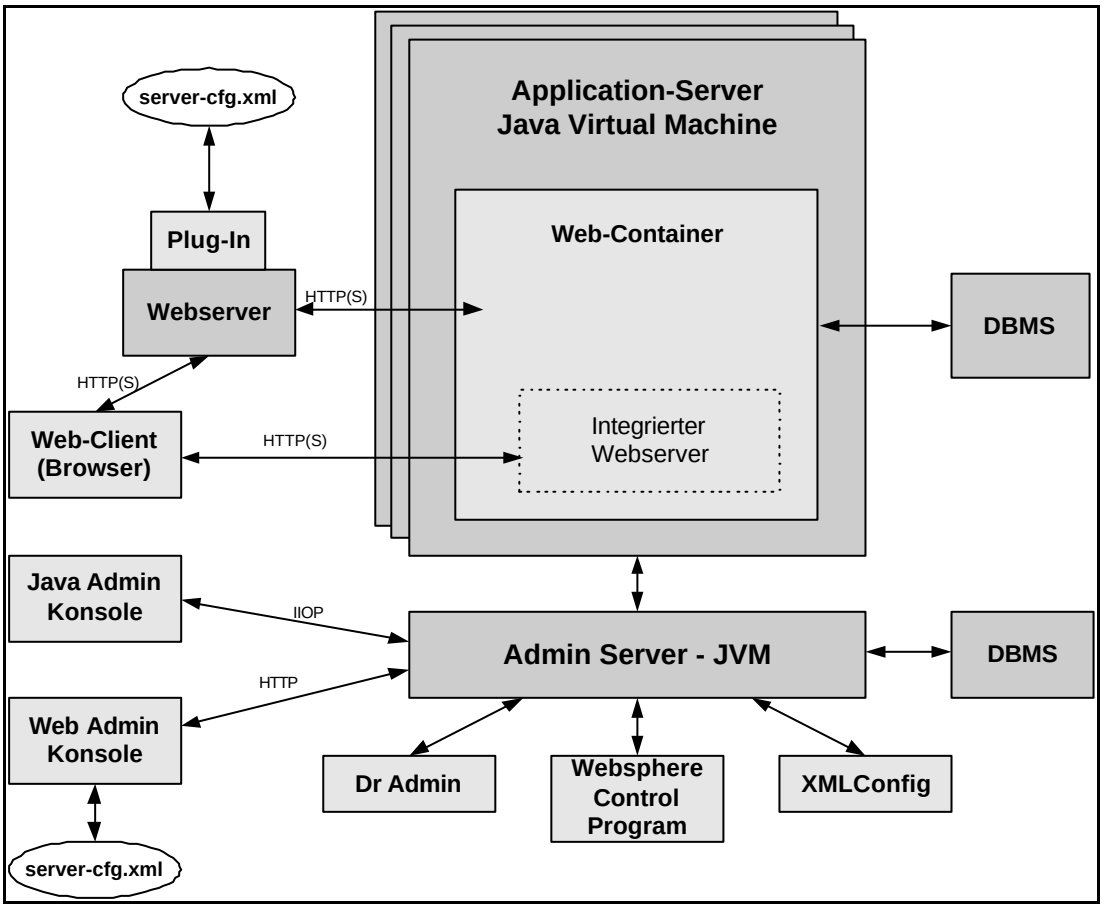


Abbildung 4-1: Hauptkomponenten des Websphere Application Server, Advanced Edition [nach EC02]

Im folgenden werden die Hauptkomponenten des Websphere Advanced Application Server näher erläutert.

Webserver und Plug-In

Für das Verarbeiten von HTTP-Anfragen, welche Ressourcen des Application Servers aufrufen, ist ein Webserver vonnöten. Die Kommunikation zwischen dem Application Server und externem Webserver ermöglicht das sogenannte Websphere HTTP Plug-In (Abbildung 4-1). Dieses Plug-In wird durch eine XML-Datei konfiguriert, in der angegeben werden kann, ob eine HTTP-Anfrage vom Web- oder Application Server verarbeitet werden soll. Es benutzt für die Kommunikation mit dem Application Server das HTTP-Protokoll. Es besteht ebenso die Möglichkeit, das sicherere HTTPS-Protokoll einzusetzen.

Das Websphere HTTP Plug-In kann mit mehreren verbreiteten Webservern, wie dem Apache Webserver, IBM HTTP Server, Microsoft IIS und Netscape iPlanet, zusammenarbeiten. [EC02, Seite 53]

Integrierter Webserver

Der Websphere Application Server besitzt einen integrierten Webserver, der über Port 9080 ansprechbar ist. Er ist lediglich für Test- und Entwicklungszwecke konzipiert und sollte nicht in Produktionsumgebungen eingesetzt werden, weil er wesentlich schlechtere Performance- und Sicherheitseigenschaften aufweist als externe Webserver. In Produktionsumgebungen sollten diese, wie zuvor erwähnt, über das Websphere HTTP Plug-In mit dem Application Server verbunden werden. [EC02, Seite 53]

Virtuelle Hosts

Virtuelle Hosts erlauben einem einzelnen Rechner, HTTP-Verbindungen an verschiedenen Ports anzunehmen und sie an ebenso viele unabhängig voneinander konfigurier- und administrierbare Applikationen weiterzuleiten. Jeder virtuelle Host besitzt einen logischen Namen und eine Liste, bestehend aus ein oder mehreren DNS-Pseudonymen, durch die er ansprechbar ist. Ein DNS-Pseudonym ist der TCP/IP Hostname und die Portnummer, getrennt durch einen Doppelpunkt in der URL einer HTTP-Anfrage an ein Servlet oder eine JSP.

Folgende Standard DNS-Pseudonyme besitzt der Websphere Application Server:

<Hostname>:80

Dieses Pseudonym benutzt den HTTP-Port des externen Webservers.

<Hostname>:9080

Hierüber wird eine Ressource über den HTTP-Port des integrierten Webservers angesprochen.

<Hostname>:443

Wird dieses Pseudonym benutzt, wird der HTTPS-Port des externen Webservers verwendet.

<Hostname>:9443

Pseudonyme dieser Form benutzen den HTTPS-Port des integrierten Webservers.

Wird eine HTTP-Anfrage an ein Servlet oder eine JSP gestellt, so wird der Servername und die Portnummer der URL mit einer Liste aller bekannten DNS-Pseudonyme für die Auswahl des richtigen virtuellen Hosts verglichen. Kann die Web-Komponente nicht gefunden werden, so wird eine HTTP-Fehlermeldung (Status: 404) an den Client zurückgegeben. [EC02, Seite 54]

4.2.2 Websphere Administrationsmodell

Das Administrationsmodell des Websphere Application Server besteht aus einer Administrationsdomäne, die einen oder mehrere Knoten umfasst. Die Knoten teilen sich ein gemeinsames Administrations-Repository in Form eines relationalen Datenbank-Systems. Ein Knoten ist ein Rechner, auf dem sich sowohl ein Application Server als auch ein Administrationsserver befinden. [EC02, Seite 60ff]
Administrations-Frontends, die sich außerhalb der Administrationsdomäne befinden können, ermöglichen die Kommunikation mit den Administrationsservern. Wie aus Abbildung 5-2 hervorgeht, besitzt die Advanced Edition dazu zwei Frontends: die Java Admin Konsole, welche das IIOP-Protokoll benutzt, und die Web Admin Konsole, die das HTTP-Protokoll einsetzt.

Der Administrationsserver
Der Administrationsserver eines Knotens stellt die Laufzeitkomponente des Websphere Administrationsmodells dar und ist für das Laufzeit- und Sicherheitsmanagement sowie die Transaktionskoordinierung verantwortlich. Weiterhin hat er die Interaktion zwischen jedem Knoten und dem Application Server in der Domäne zur Aufgabe.
Der Websphere Administrationsserver bietet Administratoren die Möglichkeit, Ressourcen und Applikationen anderer Knoten in der Domäne ebenso zu administrieren wie auf der lokalen Hostmaschine.

Administrations-Repository
Der Websphere Application Server speichert alle Konfigurationsinformationen einer Domäne zur Laufzeit persistent in einer zentralen Datenbank, Repository genannt. Standardmäßig trägt diese den Namen *WAS*. Als Datenbanken können DB2, Oracle, Informix, MS SQL Server oder Sybase verwendet werden.

Administrations-Schnittstellen
Der Websphere Administrationsserver besitzt Dienste für die Kontrolle von Ressourcen und die Ausführung von Prozessen im Administrations-Repository. Um diese Dienste, wie beispielsweise das Starten und Stoppen von Servern oder die Installation von Webapplikationen, überwachen und konfigurieren zu können, bietet der Websphere Application Server vier Schnittstellen an. Dazu gehören zwei Schnittstellen bzw. Frontends mit grafischer Benutzeroberfläche, welche bereits im

Zusammenhang mit dem Administrationsserver erwähnt wurden: der *Java Admin Konsole* und der *Web Admin Konsole*. Die restlichen beiden Schnittstellen sind kommandozeilenbasiert. Hierzu zählen *XMLConfig* und das *Websphere Control Program.*
Diese vier Schnittstellen werden im folgenden näher erläutert.

Java Admin Konsole

Die Java Admin Konsole wird primär für die Administration einer Websphere Administrationsdomäne genutzt. Alle konfigurierbaren Eigenschaften der Advanced Edition lassen sich über sie verwalten. Die Administration der gesamten Domäne durch die Java Admin Konsole kann von einem beliebigen Knoten her erfolgen, der eine Verbindung zum Administrationsserver besitzt.

Web Admin Konsole

Die Web Admin Konsole bietet die Möglichkeit, XML-Konfigurationsdateien des Websphere Application Server Advanced Edition mittels einer grafischen Benutzeroberfläche in einem Webbrowser zu editieren. Der Aufruf erfolgt über die URL: http://<Servername>:9090/admin. Die Konfiguration des Application Servers wird in der Datei server-cfg.xml gespeichert, welche beim Start der Web Admin Konsole geladen wird. Andere Konfigurationsdateien können nachgeladen werden, indem sie als Parameter an die URL angehängt werden.

Die Web Admin Konsole bietet einen geringeren Funktionsumfang als die Java Admin Konsole, da mit dem Webbrowser Funktionen nicht realisiert werden können, die für die Administration verteilter und skalierbarer J2EE-Applikationen nötig sind. [EC02, Seiten 63-64]

Die Administrationstätigkeiten im Rahmen der Diplomarbeit am Websphere Application Server wurden ausschließlich mit Hilfe der Web Admin Konsole durchgeführt, welche in Abbildung 4-2 gezeigt wird.

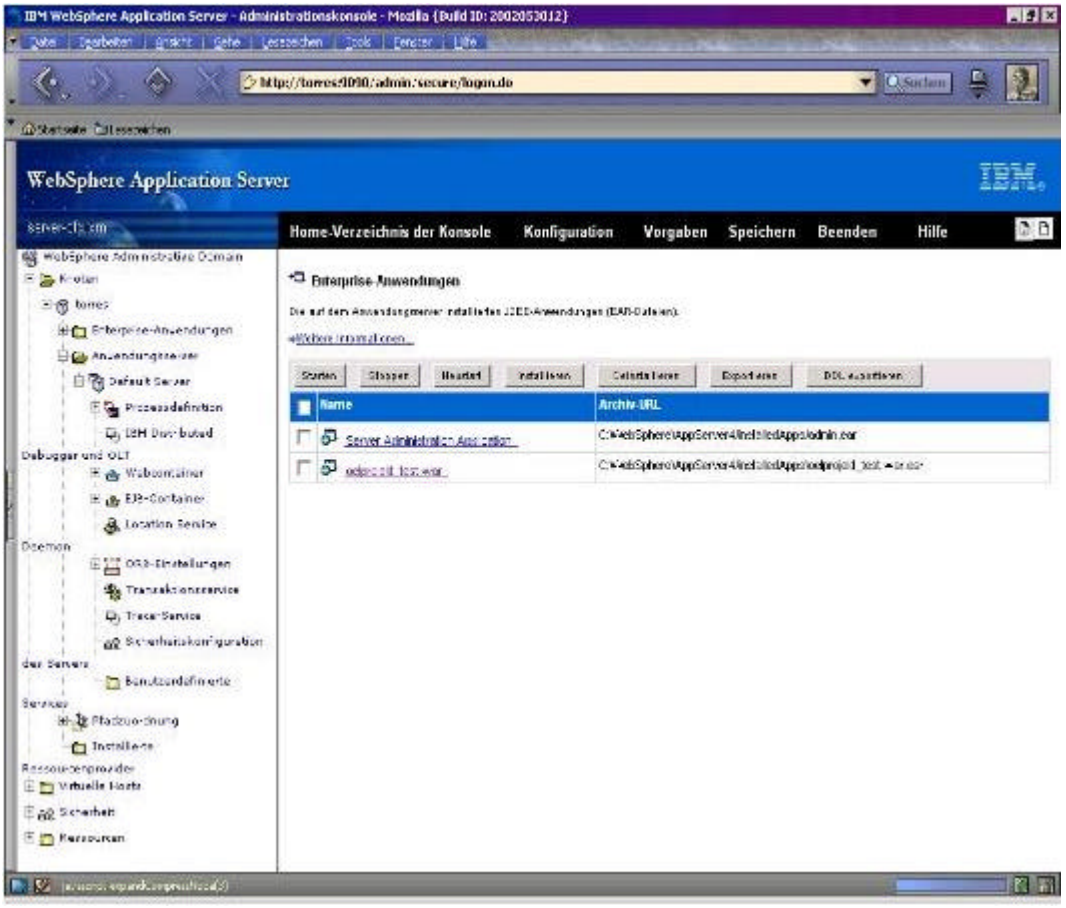


Abbildung 4-2: Die Web Admin Konsole des IBM Websphere Application Servers

Websphere Control Program

Das Websphere Control Program ist ein interaktives, kommandozeilenbasiertes Werkzeug, das die Administration von Ressourcen in einer Domäne ermöglicht, wie beispielsweise das Verwalten, Konfigurieren, Importieren und Exportieren von Konfigurationen und das Ausführen von Diagnoseprogrammen.

Der Aufruf erfolgt durch die wscp.bat – Datei unter Windows bzw. durch die wscp.sh – Datei unter UNIX-Plattformen. [EC02, Seite 64]

XMLConfig

XMLConfig stellt ein weiteres kommandozeilenorientiertes Programm dar, welches es dem Administrator ermöglicht, Websphere-Konfigurationsdaten vollständig oder partiell in bzw. aus dem Administrations-Repository zu im- oder exportieren, wobei mehrere Änderungen automatisiert erfolgen können. Desweiteren kann die Plug-In-Konfigurationsdatei plugin-cfg.xml mit entsprechenden Optionen neu generiert werden. XMLConfig arbeitet nicht interaktiv und kann keine Statusinformationen über Websphere Application Server zurückgeben. [EC02, Seiten 64-65]

Dr Admin

Dr Admin ist ebenfalls ein kommandozeilenbasiertes Werkzeug, welches bei der Fehlersuche zur Diagnose von Problemen eingesetzt werden kann.

4.2.3 Komponenten und Administration des Apache Tomcat

Im Gegensatz zum Websphere Application Server besitzt Tomcat keine vergleichbaren Administrations-Schnittstellen. Die Konfiguration und Verwaltung führt in Tomcat stattdessen über die zentrale Konfigurationsdatei server.xml, wie in Abbildung 4-3 ersichtlich ist. Die Datei spezifiziert und parametrisiert die Module, mit denen Tomcat zusammenarbeitet.
Zu den Konfigurationseinstellungen gehören u.a.:

- Zeitspanne für die Generierung von SessionID's
- Verwendung des Java-Compilers Jikes von IBM anstelle des Java-Compilers von Sun
- Akzeptierung von AJP-Anfragen von nur einer festgelegten IP-Adresse
- Ablehnung cookiebasierter Sessions

Vor dem Auslesen der server.xml verarbeitet Tomcat zuerst die Konfigurationsdatei modules.xml. mit deren Hilfe er Modulnamen die Klassen zuordnet, welche die Implementierung der jeweiligen Module darstellen. Dieses Konzept vereinfacht die Syntax der server.xml – Datei, da im Falle von aktualisierten Modulklassen lediglich die modules.xml – Datei geändert werden muss.

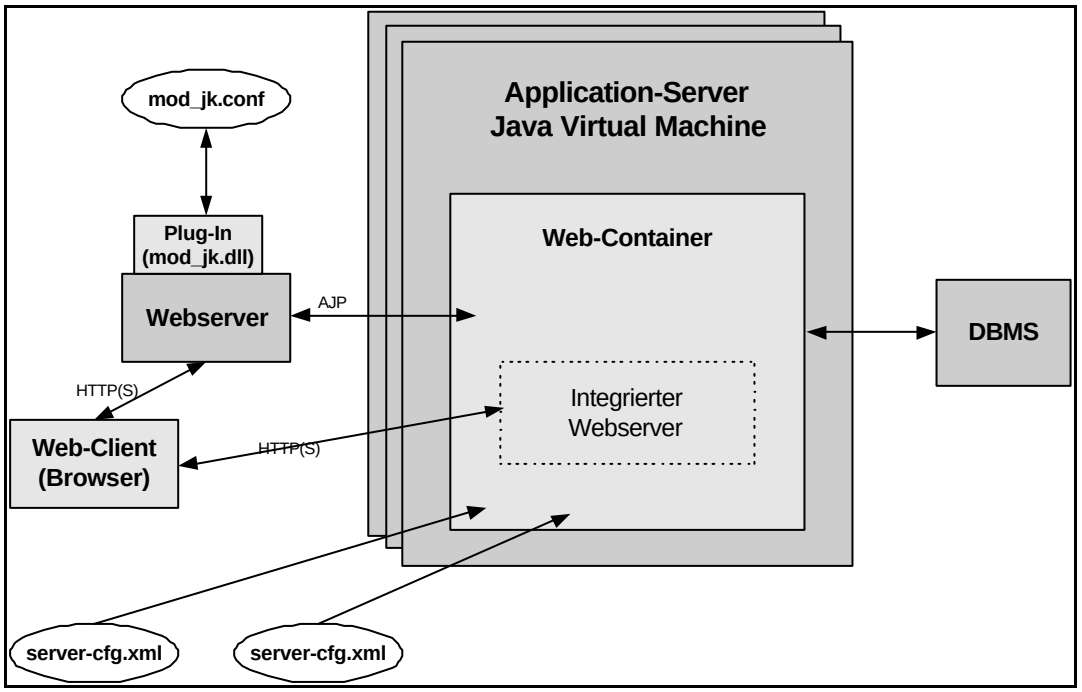


Abbildung 4-3: Apache Tomcat Hauptkomponenten

Im folgenden werden die Hauptkomponenten des Apache Tomcat näher betrachtet.

Webserver und Plug-In

Auch Tomcat bietet wie Websphere AS die Möglichkeit, externe Webserver mittels eines Plug-In's für die Client-Server-Kommunikation einzusetzen. Unterstützt werden Apache, Microsoft IIS und Netscape Enterprise, wobei für die im Rahmen der Diplomarbeit durchgeführten Messungen Apache als Webserver eingesetzt wurde.

Die Kommunikation zwischen Client und Webserver setzt das HTTP- bzw. HTTPS-Protokoll ein. Das Plug-In dagegen, dessen Implementierung bei Einsatz des Apache Webservers in Form des Modules mod_jk.dll vorliegt, benutzt für das Weiterleiten der HTTP-Anfragen und -Antworten das *Apache JServ Protocol* (AJP). Die Konfiguration des Plug-In's verläuft über die Datei mod_jk.conf, welche von Tomcat generiert werden kann und vom Webserver eingebunden wird. Der Kontrolle des Generierungsprozesses unterliegt dem ApacheConfig – Modul, welches durch die Konfigurationsdatei server.xml integriert wird. [SI02]

Integrierter Webserver

Wie der Websphere Application Server besitzt Tomcat einen über den Port 8080 ansprechbaren integrierten Webserver. Auch dieser weist wesentlich schlechtere Performance- und Sicherheitseigenschaften als separate, externe Webserver auf, da er nur für Test- und Entwicklungszwecke entwickelt worden ist. [SH02]

Virtuelle Hosts

Ebenso wie der Websphere AS besitzt Tomcat mehrere virtuelle Hosts, dessen Pseudonyme in der server.xml – Konfigurationsdatei angepasst werden können. Standardmäßig sind folgende Ports voreingestellt:

- für den HTTP-Port des integrierten Webservers: Port 8080
- für den HTTPS-Port des integrierten Webservers: Port 8443
- für den AJP12Connector, der das AJP 1.2 benutzt: Port 8007
- für den AJP13Connector, der das AJP 1.3 benutzt: Port 8009

Da Tomcat für die Kommunikation mit einem externen Webserver nicht das HTTP- bzw. HTTPS-Protokoll, sondern das eigens entwickelte AJP-Protokoll benutzt, wird entweder der AJP12Connector oder der AJP13Connector für die Verwaltung eingehender AJP-Anfragen eingesetzt. Der Einsatz des jeweiligen Konnektors hängt vom benutzten AJP-Protokoll ab: der AJP12Connector wird im

Zusammenhang mit dem älteren AJP 1.2 – Protokoll genutzt, der AJP13Connector dagegen mit dem AJP 1.3 – Protokoll.
Des weiteren werden virtuelle Hosts dazu genutzt, Web-Applikationen in Tomcat mit Hilfe des Context-Konzeptes einzubinden. Ein *Context* stellt eine Instanz einer Web-Applikation mit entsprechenden Ressourcen dar, die unter einem eindeutigen Context-Pfad erreichbar ist. Dieser Context-Pfad bildet die Verzeichnisstruktur der Web-Applikation ab. Sogenannte Context-Konfigurationsdateien ermöglichen es, weitere, spezifische Angaben über die jeweilige Web-Applikation zu machen. Wird eine Anfrage gestartet, so überprüft Tomcat die vorhandenen Context-Konfigurationsdateien auf Informationen über den virtuellen Hostnamen, der mit dem Servernamen der Anfrage übereinstimmt. [SH02]

4.3 Vergleich des Session-Managements unter Lastbedingungen

Dieser Abschnitt untersucht das Systemverhalten der für den Vergleich gewählten Application-Server unter simulierten Multi-Nutzer Lastbedingungen. Der auch Loadtesting [JH02] genannte Performance- oder Stresstest fokussiert das Session-Management der Application-Server. Entsprechend der Forderungen der Servlet-Spezifikationen [SV00] haben beide Produkte ein Sitzungskonzept implementiert, dass die Speicherung beliebiger Objekttypen erlaubt. Im Bereich von komplexen verteilten Anwendungen ist das Speichern von Daten, die in Abhängigkeit des Nutzerverhaltens stehen, von zunehmendem Interesse. Sessions spielen in diesem Anwendungsfall eine wichtiger werdende Rolle, da sie neben ihrer Fähigkeit, auch komplexeste Datenstrukturen in Form von Objekten aufnehmen zu können, auch Datenbanksysteme von Last befreien können. So können Sessions genutzt werden, um bspw. Rechte eines Nutzers bereits beim Einloggen des Nutzers komplett zu laden, um einen Datenbankzugriff zu diesem Zweck nicht bei dem Erhalt eines jeden Requests ausführen zu müssen.
Auch aus Implementierungssicht bietet sich das Arbeiten mit Sessions an, da es eine einfach zu nutzende Technik darstellt..
Der Gegenstand der folgenden Erörterungen ist die Untersuchung des Verhaltens der für das Session-Management verantwortlichen Funktionalitäten von Apache Tomcat und IBM Websphere im Bereich der Servlet-Engine.

4.3.1 Testinfrastruktur

Für die Performance-Messungen wurden Client- und Serverumgebung auf jeweils separaten Rechnern installiert, um zu gewährleisten, dass die dem Clientsystem als auch die den Serverkomponenten zur Verfügung stehenden Ressourcen voneinander entkoppelt genutzt werden können. Die

Kommunikation erfolgte über eine Peer-to-Peer 100 MBit Netzwerk-Verbindung, sodass die Resultate nicht durch von anderen Rechnern generierten Netzwerkverkehr verfälscht werden konnten.

	Client	**Server**
Prozessor	AMD Athlon 600 MHz	AMD Athlon 600 MHz
Hauptspeicher	256 MB	512 MB
Virtueller Speicher	100 MB	1 GB
Festplattengröße	20 GB	30 GB

Tabelle 11: Hardwareausstattung Testinfrastruktur

Sowohl auf dem Client- als auch auf dem Serversystem wurden *Microsoft Windows 2000 Professional* eingesetzt. Für die Simulation einer hohen Anzahl von Clients kam das Performancetool *Jmeter 1.7* der Apache Software Foundation zum Einsatz, auf Basis des Java Runtime Environment 1.3.1.
Beide Application-Server verwendeten als externen Webserver den *Apache 1.3.26*. Die Kommunikation erfolgt über Plug-Ins, die von beiden Produkten für den genannten Webserverf zur Verfügung gestellt werden. Die integrierten Webserver wurde nicht genutzt, um die Ergebnisse nicht aufgrund von wenig optimierten Webservern zu verfälschen.

4.3.2 Durchführung der Messungen

Um quantitative Aussagen über das Laufzeitverhalten der gewählten Application-Server unter Lastbedingungen treffen zu können, wurden die Performance-Metriken Verarbeitungszeit und Durchsatz gemessen. Als Durchsatz wird dabei die Anzahl der erfolgreich verarbeiteten Anfragen verstanden, als Verarbeitungszeit die durchschnittliche Dauer der Bearbeitung einer einzelnen Anfrage.

Als Grundlage der Programmierung wurde das bereits in der Beschreibung der Implementierung vorgestellte Vererbungsmodell zum Aufbau von Java Beans genutzt.

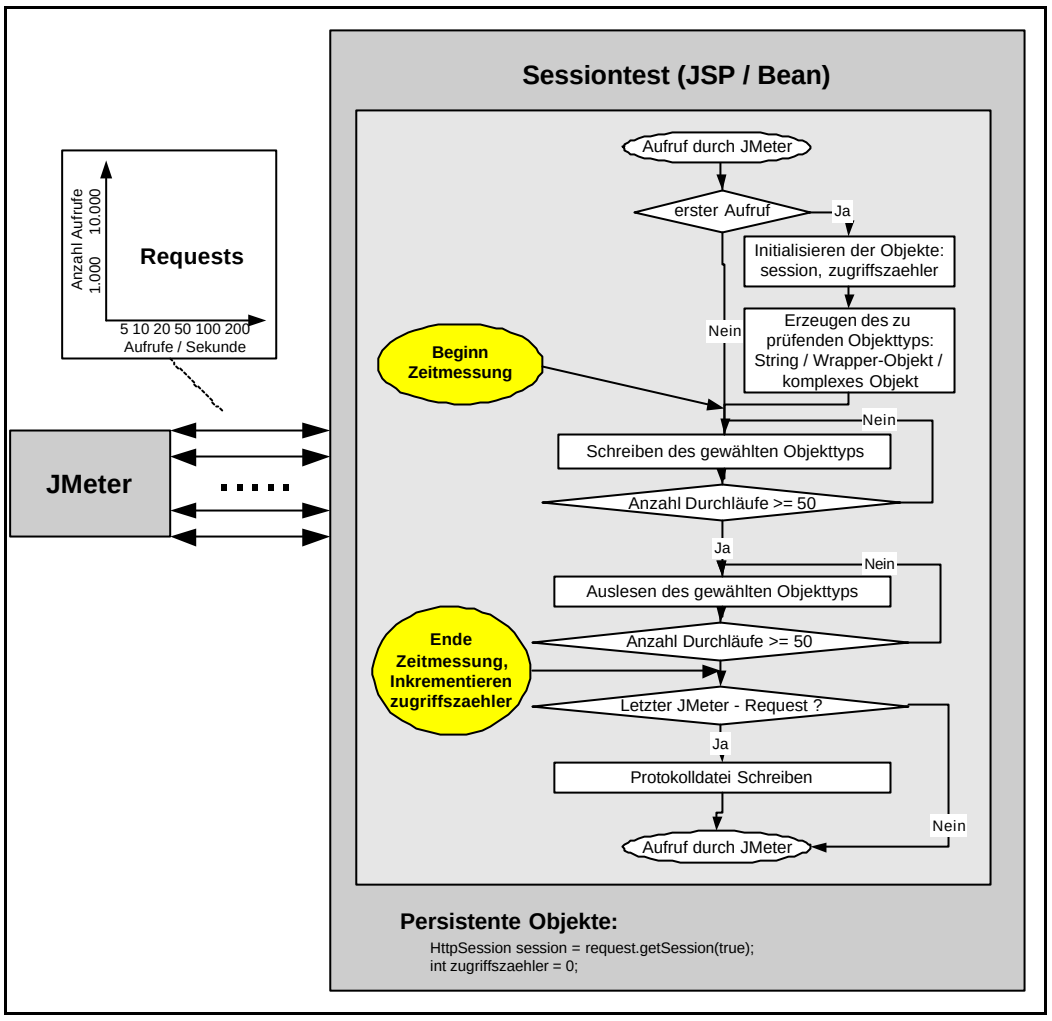


Abbildung 4-4: Zusammenhang zwischen JMeter und der messenden Java-Klasse. Die Abläufe innerhalb der Java-Klasse Sessiontest werden als Flussdiagramm skizziert.

Die Abbildung 4-4 skizziert die Vorgehensweise bei der Durchführung der Messungen. Das Programm JMeter kann so konfiguriert werden, dass es eine URL über einen definierbaren Zeitraum in einer definierbaren Anzahl pro Sekunde aufruft. Die Performancemessungen wurden in sechs Laststufen eingeteilt: 5, 10, 20, 50, 100 und 200 Anfragen pro Sekunde, wobei jede Messreihe für 1.000 wie auch für 10.000 Aufrufe erstellt wurde. Die Laststufen wurden durch einige vorgelagerte Tests ermittelt. Es handelt sich bei den Stufen um Werte, anhand derer im Test signifikante Unterschiede im Verhalten der Servlet-Engines festgestellt werden konnten. Jede der Kombinationen aus Laststufen und Anzahl der Aufrufe wurde jeweils für die drei Objekttyp durchgeführt. Als Objekttypen wurden Repräsentanten des elementaren Datentyps `String`, Instanzen von Wrapper-Klassen (`Integer`)und sehr große Objekte, die aus den Java-Datenstrukturen Vector und HashMap gebildet wurden, untersucht.

```
// Grosse Objekte in Session schreiben
for (short s = 0; s < 25; s++) {
     session.setGregorianCalHashMap();
}
for (short t = 0; t < 25; t++) {
     session.setVector();
}

// Session-Daten wieder auslesen:
for (short u = 0; u < 25; u++) {
     loginSession.getGregorianCalHashMap();
}
for (short u = 0; u < 25; u++) {
     loginSession.getVector();
}
```

Abbildung 4-5: Auszug aus der Datei Sessiontest.jsp. In eine HashMap sowie ein Vector-Objekt werden jeweils 25 Einträge geschrieben. Die Wert werden jeweils 25 und 50 Mal aus der Session ausgelesen.

Wie die Abbildung 4-5 zeigt, wurde der jeweilige Objekttyp 50 Mal in die Session geschrieben (`session.setAttribute(Sessionvariable, obj)`) und wieder ausgelesen (`objKlasse obj = (objKlasse) session.getAttribute(Sessionvariable)`).
Neben dem Datentyp `String` wurden Wrapperklassen für den primitiven Datentyp `int` in die Messungen mit einbezogen, weil sich im Fall der Beschaffungsanwendung häufiger die Notwendigkeit ergab, diese Datentypen in der Session abzulegen[12].
Die Abbildung 4-4 skizziert den grundsätzlichen Ablauf einer Messung. JMeter sendet in der jeweils konfigurierten Kombination von Laststufe und Belastungszeit Anfragen an das Servlet *Sessiontest.jsp*. Je nach übermittelten Parametern erzeugt diese Klasse die Basisobjekte zur Nutzung in der späteren Messung. Kurz vor dem Beginn der Schleife, deren Aufgabe das Schreiben der Objekte in die Session ist, wird die Systemzeit gemessen (`System.currentTimeMillis()`). Eine weitere Messung erfolgt, nachdem die Schleife der Lesevorgänge abgearbeitet wurde. Die Differenz der beiden Zeitnahmen ergibt die *Ausführungsgeschwindigkeit* der Session-Aktivitäten. Dass es sich trotz der verschiedenen Requests immer um dieselbe manipulierte Session handelt, wird erreicht, indem die Session-Instanz innerhalb des mit der JavaServer Page verbundenen Beans *SessiontestBean* deklariert wird. Da dieses Bean im JSP für den Parameter scope den Wert „session“ beinhaltet, erstreckt sich der Gültigkeitsbereich des Beans und damit auch der in ihm definierten session-Instanz auf alle Aufrufe des Messdurchlaufs. Neben der Instanz *session* wird auch die Membervariable *zugriffszaehler* sessionweit deklariert. Somit ist es möglich, die Anzahl der beantworteten Requests innerhalb des Messdurchlaufs zu zählen. Der Zeitpunkt der Inkrementierung schließt sich sofort an die letzte Zeitnahme an. Nachdem alle Requests von JMeter versendt wurden, wird eine Protokolldatei mit den

[12] In dem Bean ArtikelSuchergebnisAnzeigeBean des Funktionsbereichs Artikelsuche wird bspw. die ID der gewählten Kostenart in der Session abgelegt.

Verarbeitungszeiten und der *Anzahl der bearbeiteten Requests* erstellt. Sie stehen für die vorstehend genannten Performance-Metriken Verarbeitungszeiten und Durchsatz.

Es gibt einige Schwachpunkte des geschilderten Vorgehens zur Gewinnung von verlässlichen quantitativen Aussagen über die genannten Metriken. Exemplarisch werden drei Punkte genannt, die teilweise bereits verbessert wurden.
Bei den Versuchsaufbauten zeigte sich, dass die für die Messungen verantwortliche Methode *synchronized* deklariert sein muss. Wird dieser Punkt nicht beachtet, kommt es zu parallelen Zugriffen der durch die Requests erzeugten Threads auf die Zählervariable *zugriffszaehler*, die in falschen Werten zum Durchsatz münden. Das Problem wurde vor Durchführung der Messungen behoben.
Es fehlt derzeit die Abstimmung mit den von dem Programm JMeter gelieferten Daten. Auch JMeter verfügt über Aussagen zu Durchsatz und der Anzahl beantworteter Requests. Es misst jedoch Erfolg und Dauer des gesamten Verarbeitungswegs einer Anfrage vom Absenden des http-kodierten Requests bis zum Empfang oder Nicht-Empfang der Antwort. Es würde sich deshalb keine isolierte Betrachtung der Leistungsfähigkeit des Session-Managements realisieren lassen. Andererseits kann die Verknüpfung der beiden Messansätze durchaus interessante Probleme zutage fördern, die bei der isolierten Betrachtung unsichtbar bleiben.
Um verlässlichere Aussagen treffen zu können, müssten die Messungen insgesamt mehrfach und unter Variationen in der Rechnerkonfiguration durchgeführt werden. Auf diese Weise könnten zumindest teilweise durch das verwendete Betriebssystem erzeugte Fehler oder Verzögerungen in der Verarbeitung von Requests und der Messung des Durchsatzes offengelegt werden.

In der vorliegenden Arbeit wurden insgesamt 72 Messreihen durchgeführt (6 Laststufen x 2 Aufrufgrößen x 3 Objekttypen x 2 Application-Server). Dies ergibt eine Zahl von über 300.000 Einzelmessungen der Verarbeitungsgeschwindigkeit des Session-Managements entsprechend dem oben skizzierten Vorgehen.

4.3.3 Messergebnisse für 1.000 aufeinanderfolgende Anfragen

Der geringsten getesteten Last von 5 Anfragen pro Sekunde zeigen sich beide Application-Server gewachsen. Sowohl bei der Session-Manipulation von String- wie auch Wrapper- und komplexen Objekten werden alle Anfragen beantwortet. Die Antwortzeiten liegen abhängig vom Objekttyp in einem Korridor von 9 ms – 23 ms (String- und Wrapperobjekte) bzw. von 24 ms – 38 ms bei den komplexen Objekten. Bei einer Erhöhung der Last zeigen die beiden Application-Server jedoch ein grundsätzlich verschiedenes Verhalten. Die Abbildung 4-6 stellt die unterschiedlichen Reaktionen auf steigende Lasten in einem Diagramm dar. Das Diagramm beschreibt die Differenz der erfolgreich beantworteten Anfragen zwischen Websphere und Tomcat dar. Bis zu einer Frequenz von 20 sekündlichen Anfragen kann der Websphere Application-Server einen deutlich höheren Prozentsatz an

Session-Operationen erfolgreichen ausführen. Der größte Unterschied liegt im Bereich der Wrapper-Objekte, von denen er bei 1.000 Anfragen 989 erfolgreich bearbeitet. Tomcat hingegen bricht die Abarbeitung bei 70 von 1.000 Anfragen ab.[13]

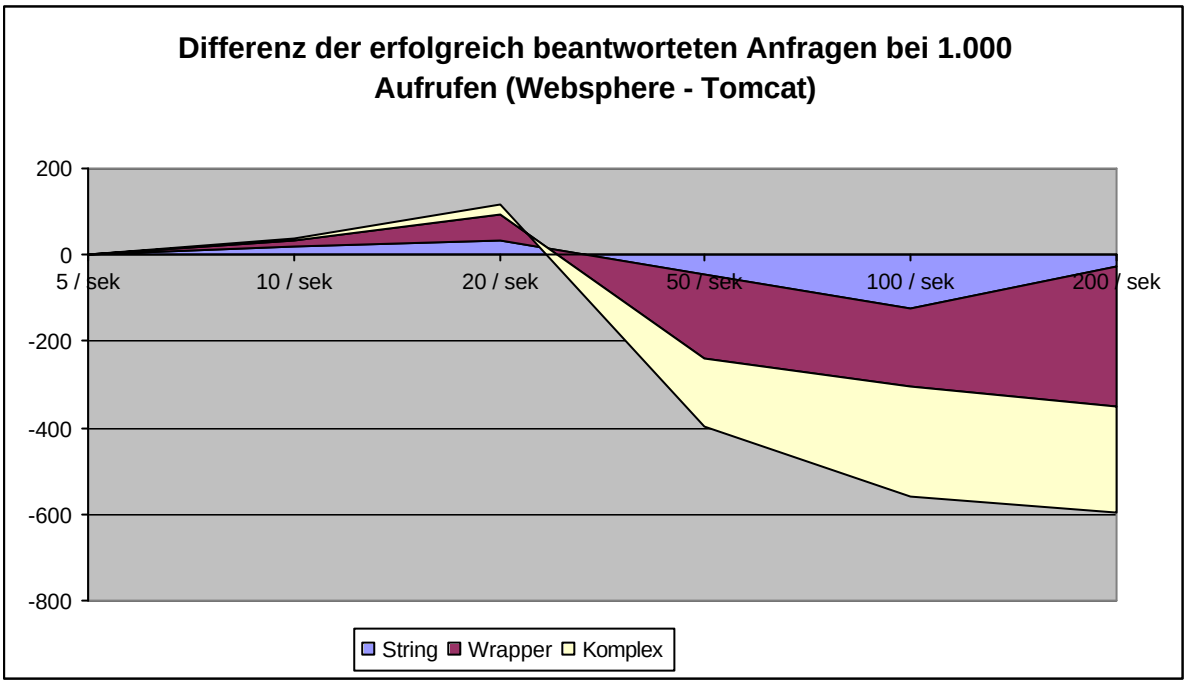


Abbildung 4-6: Vergleich der erfolgreich bearbeiteten Anfragen. Das Diagramm zeigt pro Objekttyp die Differenz (Websphere - Tomcat) der erfolgreich bearbeiteten Anfragen von insgesamt 1.000 Aufrufen (logarithmische Darstellung).

Bei der folgenden Frequenzerhöhung auf 50 Anfragen pro Sekunde invertiert sich unerwartet das Verhältnis. Auch hier zeigen die Wrapper-Objekte wieder die größten Unterschiede. Tomcat kann bei dieser Frequenz mit 889 immerhin 197 Anfragen mehr erfolgreich bearbeiten als Websphere. Die Tendenz setzt sich bei einer weiter steigenden Anzahl von Aufrufen fort. Während die Dokumentation der Version 4.0 des Websphere Application Servers beschreibt, wie dieses Verhalten über Konfigurationseinstellungen gesteuert werden kann, ist in der Dokumentation zu der eingesetzten Version 3.5 kein Hinweis auf möglicherweise vorgegebene Einstellungen zu finden. Dass der Websphere Application Server tatsächlich größere Probleme bei den dargestellten Aufruffrequenzen hat, wird auch in [GH00, Seite 4] dargestellt und mit dem Hinweis bedacht, speicherintensive Session-Objekte alternativ in einer Datenbank zwischen zu speichern. Der Artikel beschreibt ebenfalls die starke Abhängigkeit der Leistungsparameter von dem zur Verfügung stehenden Hauptspeicher.

Im Unterschied zu dem IBM-Produkt zeigt Apache Tomcat ein weniger überraschendes Verhalten. Der Durchsatz nimmt regelmäßig ab und erleidet keinen Einbruch. Tomcat verliert auch bei den getesteten, sehr hohen Lasten nicht mehr als 20% der Anfragen, während der Wert des Kontrahenten Websphere hier bis zu 44% steigt.

[13] Eine tabellarische Darstellung aller Messwert inklusive ihrer graphischen Darstellung befindet sich im Anhang 3.C

4.3.4 Messergebnisse für 10.000 aufeinanderfolgende Anfragen

Die Tendenz der in vorstehenden Absatz festgestellten verschiedenartigen Reaktionen auf die verschiedenen Lasten setzt sich erwartungsgemäß bei der Messreihe von 10.000 Anfragen fort.

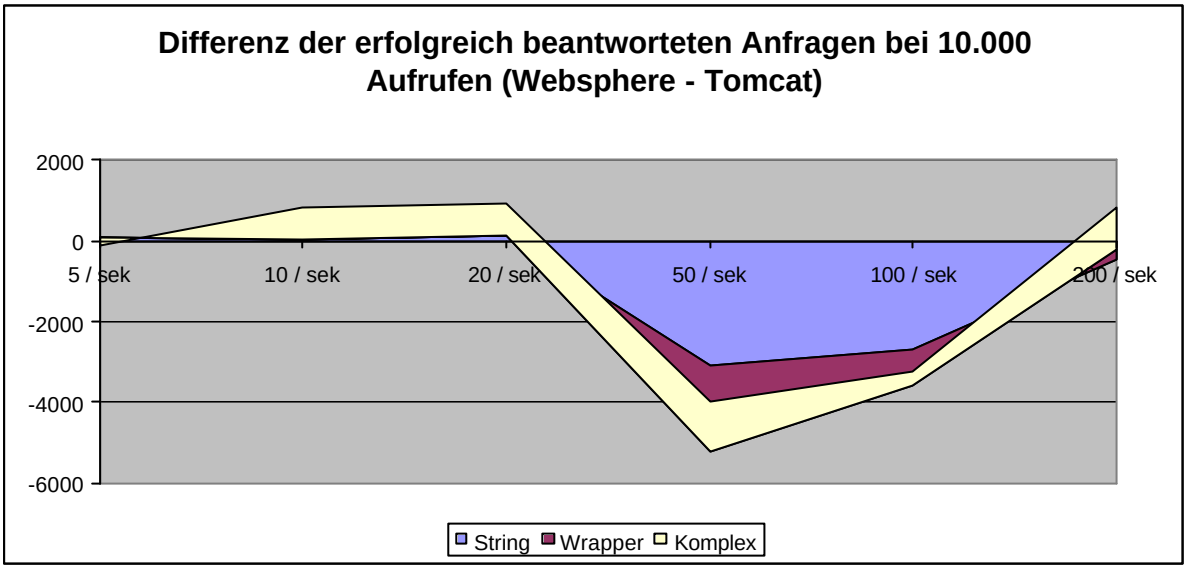


Abbildung 4-7: Vergleich der erfolgreich bearbeiteten Anfragen. Das Diagramm zeigt pro Objekttyp die Differenz (Websphere - Tomcat) der erfolgreich bearbeiteten Anfragen von insgesamt 10.000 Aufrufen (logarithmische Darstellung).

Es kann jedoch konstatiert werden, dass sich die erwartete Deutlichkeit im Durchsatz der beiden Application-Server bis 20 Anfragen pro Sekunde nur bei den komplexen Objekten wiederspiegelt. Bei String- wie auch Wrapper-Instanzen fällt der Unterschied geringer aus. Die Messungen bei 5 Anfragen pro Sekunde weisen nur bei den String-Klassen einen kleinen Vorteil von 1% mehr verarbeiteten Aufträgen aus. Dahingegen verliert Websphere im Bereich der komplexen Objekte gegenüber Tomcat 227 mehr Anfragen. Bei den nächsten Anfragefrequenzen von 10 und 20 Aufträgen pro Sekunde kehrt sich die Situation wieder um, wie die Abbildung 4-7 zeigt. Wie bei genauerer Betrachtung der Abbildung 0-8 und der Abbildung 0-9 des Anhangs deutlich wird, bewegen sich die absoluten Antwortzahlen des Session-Managements bereits zu diesem Zeitpunkt in einem deutlich schlechteren Bereich. Schon bei 10 Anfragen pro Sekunde werden von den Application-Server nur mehr 60 – 70% der komplexen Anfragen erfolgreich bearbeitet. Andererseits zeigen sich beide Testkandidaten im Bereich der String- und Wrapper-Klassen in ihrem Element. Von den 6 Messwerten für die beiden Objekttypen und die Frequenzen 5, 10 und 20 Anfragen pro Sekunde hat Websphere viermal alle 10.000 Anfragen erfolgreich ausgeführt. Nur knapp darunter befindet sich der Apache Tomcat.

Auch die Antwortzeiten spiegeln die Sonderrolle der komplexen Objekte für das Session-Management wider. Die durchschnittliche Antwortzeit für 10 Anfragen pro Sekunde liegt im Fall des Application-Servers Apache Tomcat bei 10.000 Anfragen bei dem Achtfachen des Werts für 1.000 Anfragen. Bei

Websphere ist das Verhältnis trotz absolut etwas geringerer Antwortzeiten sogar bei dem Neunfachen. Dieses erstaunliche Ergebnis erklärt sich aus sprunghaft zunehmenden Antwortzeiten, wie sie beispielhaft für den Tomcat Application-Server in Abbildung 4-8 dargestellt werden.

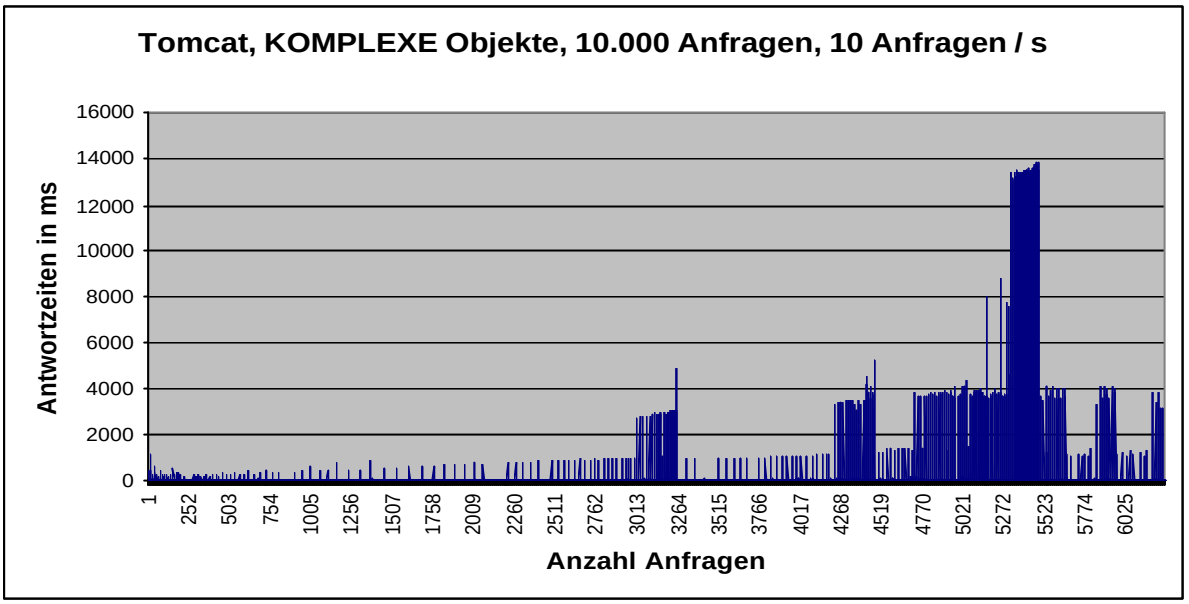


Abbildung 4-8: Antwortzeiten von Apache Tomcat bei der Session-Verarbeitung von komplexen Objekten (10.000 Anfragen, 10 Anfragen / Sekunde)

Im Vergleich dazu stellt die folgende Abbildung 4-9 die Antwortzeiten der 10.000er Messreihe bei 10 Anfragen pro Sekunde für die einfacheren Wrapper-Objekte dar.

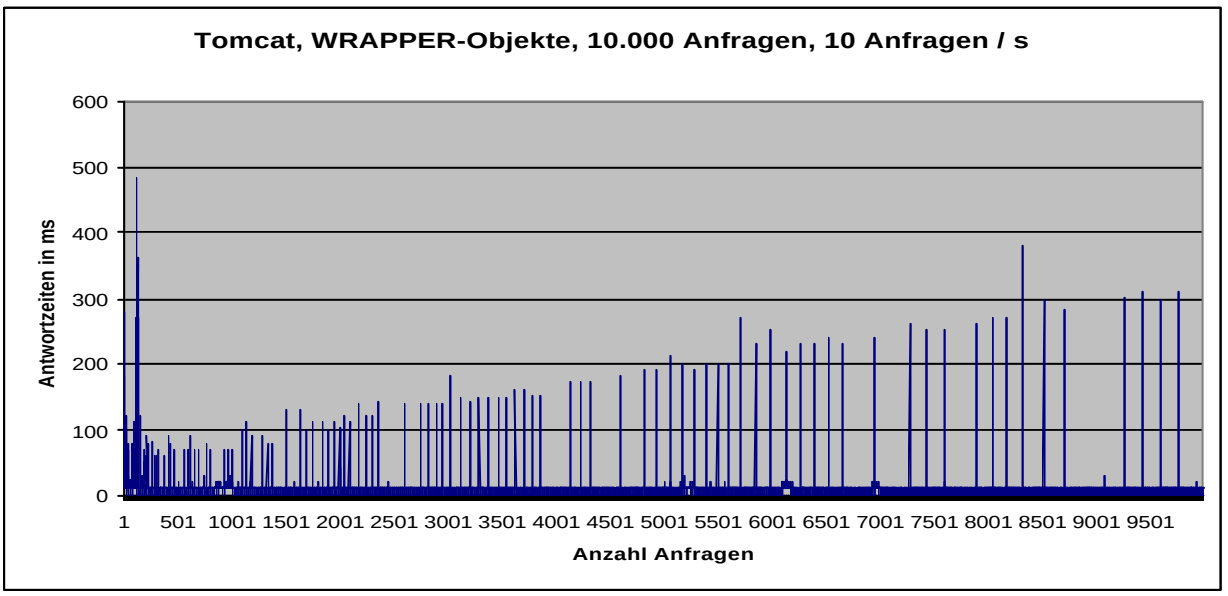


Abbildung 4-9: Antwortzeiten von Apache Tomcat bei der Session-Verarbeitung von Wrapper-Objekten (10.000 Anfragen, 10 Anfragen / Sekunde)

Die Abbildungen machen deutlich, dass es bei der Verarbeitung von komplexen Objekten ab etwa 3.000 ausgelösten Anfragen erstmals zu einem sprunghaften Anstieg der Antwortzeiten kommt. Das Antwortzeitverhalten wiederholt sich ab etwa 4200 Anfragen mehrmals und erreicht Spitzenwerte bei etwa 5200 Anfragen, wo die Antwortzeiten bei bis zu 14 Sekunden pro Anfrage liegen. Die Verarbeitung von Wrapper-Klassen zeigt hingegen ein erwartetes Bild. Regelmäßig liegen die Antwortzeiten bei 10 bis 20 ms. Unterbrochen werden sie alle etwa 60 – 70 Anfragen durch eine Antwortzeiten-Spitze, die sich von etwa 100 ms stetig bis zu etwa 400 ms steigert. Eine Erklärung hierfür ist nicht in der Dokumentation zu finden. Man kann jedoch davon ausgehen, dass es sich bei diesen lokalen Maxima um Speicherreorganisationen der Session-Engine handelt, denn bei jeder Anfrage wird eine neue Session geöffnet, für die Speicherplatz bereitgestellt werden muss. Da die Reorganisation jeweils mehr Objekte verwalten muss, kann es zu der stetigen Steigerung der Extrema kommen.
Diese Argumentation erklärt jedoch nicht, wie es zu den langandauernden Einbrüchen bei der Verarbeitung von komplexen Objekten in Sessions kommt, die sich sowohl bei Apache Tomcat als auch bei Websphere zeigen. Möglicherweise müssen aufgrund des hohen Speicherbedarfs der Sessiondaten umfangreiche Speicherallokationen in der Java VM durchgeführt werden, die auch zu einer Nutzung der langsamen Auslagerungsdatei der Windows-Systeme führt. Es könnte auch sein, dass die Algorithmen der Session-Verwaltung bei den getesteten hohen Lasten ineffizient arbeiten. Die Gründe für das Verhalten können im Rahmen der vorliegenden Arbeit nicht ermittelt werden.
Es bleibt festzuhalten, dass das Antwortverhalten der Application-Server unter Dauerbelastung wesentlich vom Typ der verwalteten Session-Daten abhängig ist.

4.4 Kernaussagen des Vergleichs

Die vorstehende Beschreibung der von der Application-Servern mitgebrachten Entwürfe für die Administration ihrer Komponenten zeichnet ein klares Bild. Der Websphere Application Server bietet ein durchdachtes Konzept zur Administration. Die Definition von Knoten ermöglicht das verteilte Überwachen und Konfigurieren von Application-Servern und ihrer Applikationen.
Tomcat hat kein vergleichbares Administrationsmodell. Werden alle wichtigen Konfigurations- und Administrationsaufgaben, die den Application Server betreffen, im Websphere Administrationsmodell durch einen Administrationsserver durchgeführt, so wird dies bei Tomcat manuell in der zentralen Konfigurationsdatei server.xml vorgenommen. Anstelle eines Administrations-Repository's werden alle Einstellungen in dieser oder anderen Konfigurationsdateien im XML-Format gespeichert. Die Konfiguration wird erschwert, da der Administrator die entsprechenden Einstellungen im einzelnen im XML-Format kennen muss. [SH02]

Der Performance-Vergleich der Application-Server arbeitet ebenfalls deutliche Unterschiede heraus.

Das Produkt von IBM positioniert sich als performant und zuverlässig in einem Bereich bis 20 Anfragen pro Sekunde auf Basis der beschriebenen Hard- und Softwareausstattung. In dem Bereich von mehr Anfragen pro Zeiteinheit zeigt Tomcat das insgesamt berechenbarere Verhalten. Während Tomcat einen stetige Verringerung der erfolgreich verarbeiteten Aufrufe bei zunehmender Last pro Sekunde zeigt, lässt sich bei dem Websphere Application Server ein Einruch verzeichnen.
Interessant sind auch die unterschiedlichen Resultate bei der Verarbeitung von verschiedenen Objekttypen gerade bei lang anhaltenden Lasten. Beide Produkte zeigen hier selbst bei noch geringen Lasten pro Zeiteinheit große Schwierigkeiten in der Verarbeitung von komplexen Sessiondaten, während die Verwaltung von einfacheren Objekten in zu erwartender Weise funktioniert.

5 Ansatzpunkte zum Einsatz eines Data Warehouse

Das Dezernat 1 definiert als einen wichtigen Punkt der Anforderungen an die entwickelte Beschaffungsanwendung die *Erfüllung der steigenden Anforderungen zur Analyse des Mitteleinsatzes und der Bereitstellung von Daten für die Kosten/Leistungsrechnung der Universität*[14].
Zum Zeitpunkt der Durchführung der im Kapitel 2 durchgeführten Analysen gab es keine klaren Vorstellungen, welche Informationen der Kosten/Leistungsrechnung der Universität zur Verfügung gestellt werden müssen. Jedoch haben die Nutzer des Sachgebiets 12 sowie der Einrichtungen ein wachsendes Interesse an Auswertungen, die in dem Funktionsbereich Listen der Anwendung angeboten werden[15]. Die hierbei gesammelten Informationen über die Sichtweise des Dezernats 1, in dessen Bereich auch die Durchführung der Kosten/Leistungsrechnung fällt, mündet in den vorliegenden Vorschlag zu dem Entwurf eines Data Warehouse. Auch aus technischer Sicht ist der Einsatz eines Data Warehouse empfehlenswert, um die Belastung des Systems durch Auswertungen, die jeweils in Echtzeit über große Mengen auch historischer Daten durchgeführt werden müssen, zu verringern.

5.1 Grundlagen

Dem Aufbau eines Data Warehouse geht eine intensive Beschäftigung mit der Frage voraus, ob sein Einsatz das erstrebte Ziel einer übergreifenden Auswertungsmöglichkeit aus verschiedenen Perspektiven erreichen kann. Legt man die Begriffsdefinition von W.H. Inmon zugrunde:

> *A Data Warehouse is a subject-oriented, integrated, non-volatile, and time variant collection of data in support of management decisions.[WI96]*

[14] siehe Abschnitt 1.2.2

[15] siehe hierzu Abschnitt 3.3.2.9

eröffnet sich die Grundlage für eine Einschätzung dieser Frage.

Die genannte *Themenorientiertheit* ist der grundsätzliche Ansatzpunkt des Controllings der Universität. Die gewünschten Informationen beziehen sich bspw. auf aggregierte Werte zu Fakultäten, Bestellungen, Nutzern, etc. Die Integration erfordert im Fall der Beschaffungsanwendung keine weiteren Aktivitäten, da die Informationen aus einer homogenen Datenquelle kommen. Dennoch ist die Fähigkeit eines Data Warehouse, Informationen heterogener Datenquellen zu *integrieren*, für eine mögliche Erweiterung des Einsatzfelds eines Data Warehouse von großem Interesse. Die Kosten/Leistungsrechnung hat im Laufe des Jahres 2002 ein neues Abrechnungssystem der Firma HIS zum Einsatz gebracht, dessen Daten in das genannte Data Warehouse integriert werden könnten. Aus Sicht des Dezernats 1 hat dies den Vorteil einer globalen Sicht auf die Beschaffungsvorgänge, die beteiligten Einrichtungen, Nutzer, etc. und nicht nur auf die durch die Anwendung derzeit abgedeckten Beschaffungsvorgänge der in der Analyse (Kapitel 2) genannten Kostenarten. Eine weitere Eigenschaft von Data Warehouses ist die *Bereitstellung einer stabilen, persistenten Datenbasis*, deren Daten im i.a. nicht mehr geändert werden. Diese Eigenschaft und die Tatsache, dass Data Warehouses auch historische Daten führen, können die Beschaffungsanwendung in einem wichtigen Punkt entlasten, nämlich der *Auswertung von historischen Daten*. Durch die intensive Nutzung der Beschaffungsanwendung wachsen die Datenstämme des zugrundeliegenden Datenbanksystems von Microsoft mit hoher Geschwindigkeit an. Die Belastung des Systems wird durch Auswertungen, die über nicht aggregierte und sich nicht mehr verändernde Daten iterieren, stark belastet. Ein Data Warehouse könnte die für mögliche Auswertungen notwendigen Daten übernehmen und dauerhaft speichern, sowie die erforderlichen Auswertungen ausliefern, ohne dass die Datenbasis der Beschaffungsanwendung berührt würde.

Insgesamt ergibt sich aus den vorstehend genannten Eigenschaften eines Data Warehouses eine Reihe von Vorteilen für die Beschaffungsanwendung sowie die Offenheit zur Integration weiterer Datenquelle und Auswertungsanforderungen.

Die folgenden Vorschläge für den Entwurf eines Data Warehouse beziehen sich nur auf die Beschaffungsanwendung und nicht auf die vom Controlling der Universität Leipzig möglicherweise in anderen Softwareprojekten geforderten Auswertungen. Der Entwurf beschreibt mögliche Dimensions- und Faktentabellen und erläutert ihre Beziehung zu den im Datenbankschema der Anwendung aufgeführten Tabellen.

5.2 Vorüberlegungen zu einem Entwurf

Die Abbildung 5-1 skizziert in Form eines Starschemas die Fakten- und denormalisierten Dimensionstabellen sowie die Kennzahlen. Neben dem Wert ist auch die Menge, in der ein Artikel bzw. Gruppierungen von Artikeln bestellt wurden, für das Controlling von Interesse. Über den

Mengenwert können unabhängig vom Wert des Artikels Aussagen zu seiner Umschlagshäufigkeit gewonnen werden.

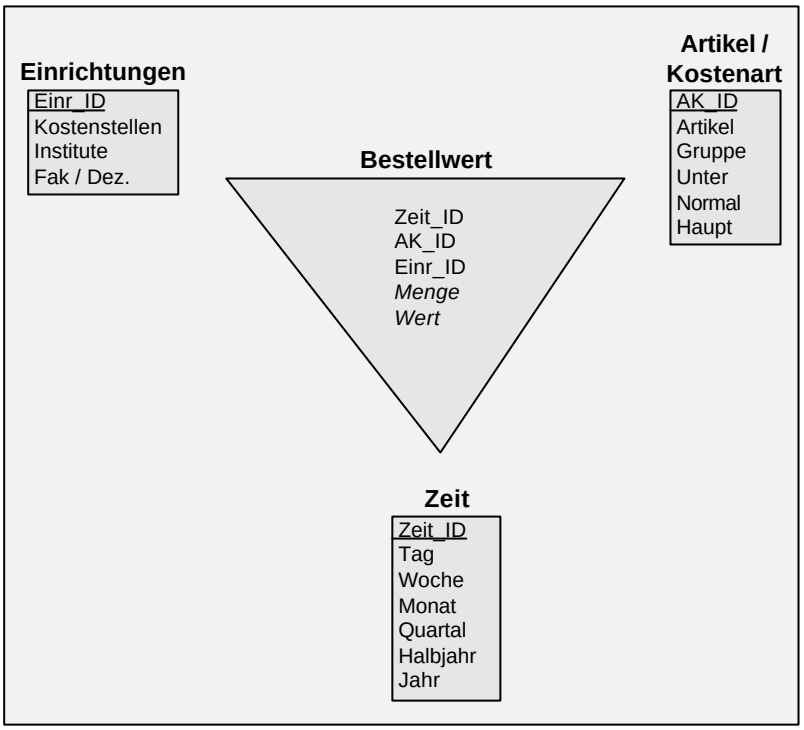


Abbildung 5-1: Entwurf eines Starschemas für die Beschaffungsanwendung

Artikel gruppieren sich entsprechend dem bereits in der Implementierung geschilderten Kostenartensystem in mehrere hierarchische Gruppen. Der Wert *Gruppe* bezeichnet die sechsstelligen internen Kostenarten, auch als Artikelgruppen bezeichnet, während die Bezeichnungen *Unter*, *Normal* und *Haupt* den einander jeweils übergeordneten dreistelligen, zweistelligen und einstelligen Kostenartbezeichnungen entsprechen wie sie im Abschnitt 3.3.2.6 im Funktionsbereich der Konfiguration näher beschrieben werden. Die Werte der Dimensionstabelle *Einrichtungen* entsprechen der an der Universität Leipzig gebräuchlichen Zuordnung von Kostenstellen zu Instituten, die wiederum Fakultäten zugeordnet sind. Es ist im Bereich der Dezernate auch möglich, dass keine Pendant zu den Instituten existiert. Die Dimensionstabelle *Zeit* beinhaltet die Standardwerte.
Die typischen Fragestellungen, die sich in den Anforderungen an die erstellten Listen manifestieren, lauten:

> *Wie viel Geld wurde im Jahr 2002 für den Einkauf von Papiererzeugnissen ausgegeben ?*
> *Welche Fakultäten haben den höchsten Verbrauch an allgemeinem Bürobedarf im ersten Quartal 2002?*
> *Welche Artikel wurden im Verlauf des ersten Halbjahres am häufigsten / seltensten bestellt ?*

Fragen dieses Typs können auf Basis des vorgeschlagenen Entwurfs beantwortet werden.
Erweiterungen könnten in dem Bereich der Nutzeranalyse zu finden sein, bzw. der Frage, welche Nutzer innerhalb eines bestimmten Zeitraums wenig oder viel gekauft haben. Bis zum jetzigen

Zeitpunkt wurde diese Granularität in den Anforderungen nicht definiert, weil es für das universitätsweite Controlling irrelevant ist. Andererseits können die Fakultäten und Institute ein so erweitertes Data Warehouse nutzen, um Transparenz bei den Einkäufen der bestellberechtigten Nutzer zu erhalten. Da Nutzer für mehrere Kostenstellen bestellberechtigt sein können, bietet es sich an, einen weiteren Wert *Nutzer* in die Dimensionstabelle der Einrichtungen zwischen *Kostenstellen* und *Instituten* aufzunehmen. Es muss dann jedoch geklärt sein, dass ein Nutzer nicht für die Kostenstellen von verschiedenen Instituten Bestellungen auslösen kann.

5.3 Tools des Microsoft SQL Server 2000

Der Microsoft SQL Server 2000 bietet insbesondere zwei Tools an, die beim Erstellen eines Data Warehouse nützliche Werkzeuge sind. Der *Data Transformation Service (DTS)* und *die Analysis Services*.

Das Feature *DTS* dient im Wesentlichen dazu, mit Hilfe graphischer Werkzeuge Daten von einer Datenquelle in eine andere zu transformieren. Dabei kann der Begriff Transformieren sowohl ein reines Kopieren bedeuten wie auch eine mehr oder minder starke Manipulation der Daten auf dem Weg von der Datenquelle zum Datenziel. Der DTS ist konzipiert, Transformationen zwischen heterogenen Datenquellen und –zielen zu organisieren und auszuführen. Er ermöglicht auch aber auch die Ausführung von Aufgaben, die um die reine Datentransformation von der Quelle zum Ziel herumgelagert sind. So kann bspw. die Ausführung von SQL-Skripten beliebiger Art oder der Start ausführbarer Dateien in eine DTS-Gesamtaufgabe integriert werden.

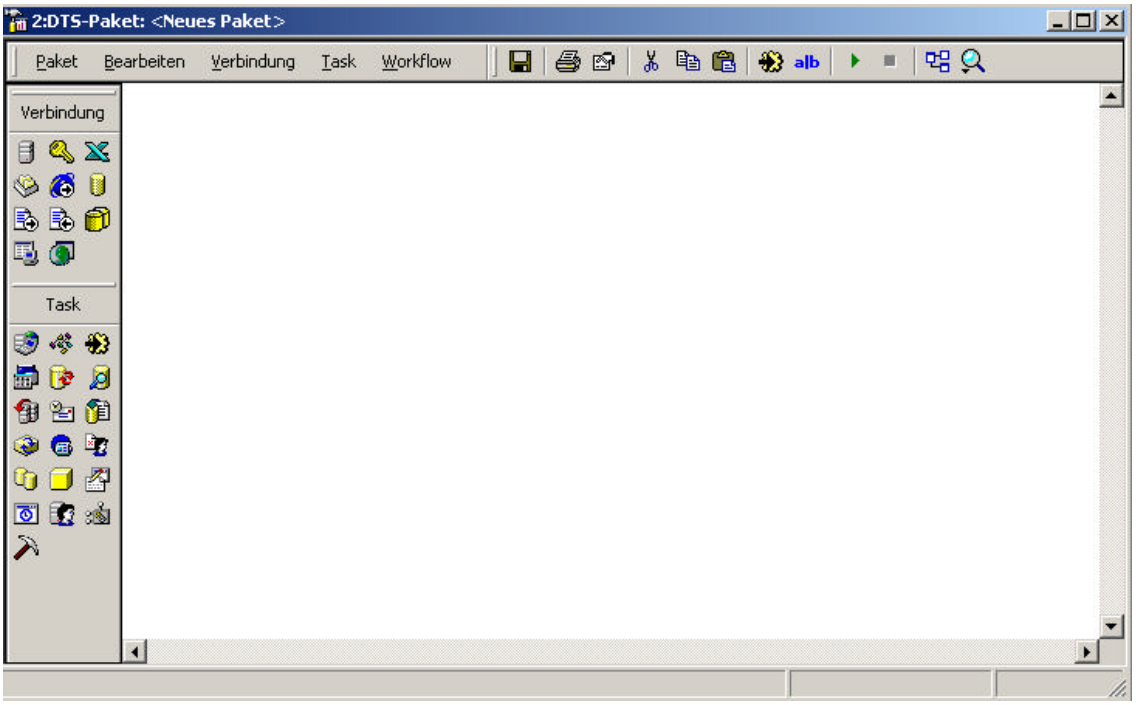


Abbildung 5-2: Fenster des DTS-Designers für ein leeres neues Paket

Eine DTS-Gesamtaufgabe kann aus beliebig vielen Teilaufgaben – den Tasks – zusammengesetzt sein. Ihre Ausführungsreihenfolge wird durch Workflows bestimmt, die zwischen den einzelnen Tasks definiert werden können.

Analysis Services sind dazu ausgelegt, Daten, die zunächst in relationaler Form angeboten werden, in multidimensionale Daten zu überführen und zu verwalten. Die multidimensionalen Daten werden in einem Cube organisiert und gespeichert. Voraussetzung für das Erstellen eines Cubes sind die zeitlich vorgelagerte Definition der Datenbank, die Dimensionen und Cubes enthält, der Datenquelle, die bspw. mit dem DTS aufbereitet wurde und der Dimensionen des Cubes.

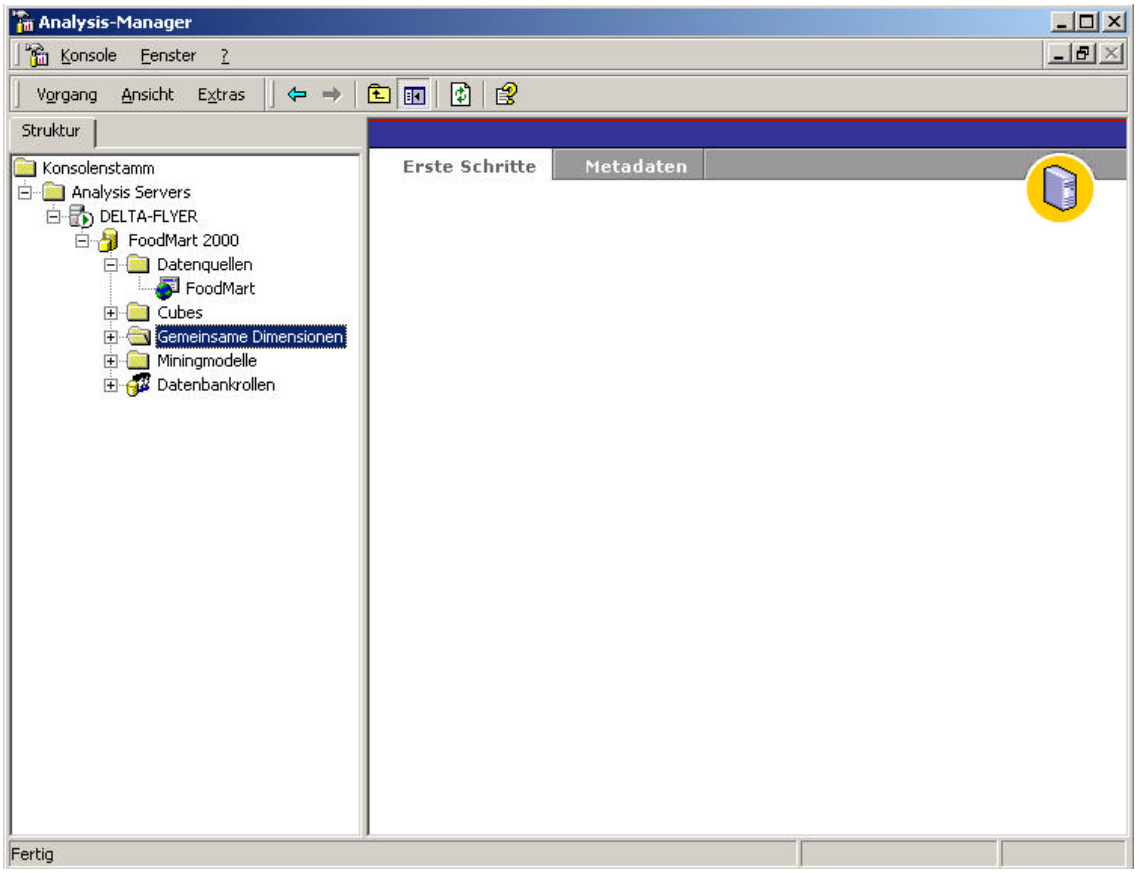


Abbildung 5-3: Übersicht der vom Analysis Server verwalteten Bestandteile eines Data Warehouse anhand der Demoanwendung FoodMart 2000

Die vorstehenden Ausführungen sind erste Ansatzpunkte zur Erweiterung der Beschaffungsanwendung um ein Data Warehouse. Die inhaltlichen Ansatzpunkte sind deutlich zu erkennen und der bereits für die Datenbasis eingesetzte Microsoft SQL Server 2000 bietet die notwendigen Features ohne den Erwerb zusätzlicher Module bereits mit an.

6 Zusammenfassung

6.1 Ergebnisse

Im Rahmen der vorliegenden Arbeit wurde eine Anwendung zur Unterstützung der Bestellprozesse zwischen den Einrichtungen und dem zentralen Lager der Beschaffungsabteilung (Sachgebiet 12) des Dezernats 1 der Universität Leipzig entwickelt. Die Tätigkeiten umfassten die Analyse, Konzeption, Entwicklung und Begleitung der Inbetriebnahme einer Beschaffungsanwendung bis zum produktiven Einsatz.

Die bis zum jetzigen Zeitpunkt bekannten Reaktionen nach der Einführung der Anwendung im April 2002 sind sehr positiv und lassen erkennen, dass die Anwendung intensiv genutzt wird. Nachdem die in den vorstehenden Kapiteln beschriebenen Grundfunktionalitäten bereitgestellt waren, wurde seitens der Nutzer bereits eine Reihe von Vorschlägen zu Erweiterungen der Anwendung gemacht.

Die entwickelte Anwendung stößt auch bei der Firma HIS GmbH (*HIS Hochschul-Informations-System GmbH*)[16] auf großes Interesse. Die HIS GmbH entwickelt Software für die verwaltungstechnischen Belange von deutschen Universitäten und kann in diesem Segment als Marktführer betrachtet werden. Erste Kontakte, die der Betreiber der Anwendung im ASP-Betrieb, die get AG, Leipzig, mit der HIS GmbH aufgenommen hat, zeigten die Bereitschaft für eine Kooperation in den von der Beschaffungsanwendung abgedeckten Prozessbereichen.

Die get AG, Leipzig, stellt die Anwendung als Application Service Provider zur Verfügung und sichert den weiteren Betrieb sowie die fortgesetzte Entwicklung der Anwendung ab.

Die Untersuchungen zu den Leistungsparametern der untersuchten Application-Server zeigen die unterschiedlichen Ausrichtungen der Produkte. Das open-source Produkt Apache Tomcat 3.3 zeigt insgesamt betrachtet eine solide Leistung. Die festgestellten Schwächen in der Verwaltung von komplexen Serssion-Objekten bei andauernd hoher Last, die zur Erzeugung von einigen tausend parallelen Session führt, fällt bei dieser Bewertung nicht stark ins Gewicht, da es sich um einen selten auftretenden Extremfall handelt. Hinzu kommt, dass auch das IBM-Produkt Websphere, Version 3.5 mit Schwächen in diesem Bereich zu kämpfen hat. Die Messdaten aus den Session-Tests lassen vermuten, dass Websphere bei einer optimierten Konfiguration bis zu einem erkennbaren Grenzwert sehr zuverlässig arbeitet, während Tomcat auch bei geringeren Lasten einzelne Anfragen nicht abarbeitet. Das IBM-Produkt hat hier klare Vorteile. Einen weiterer Pluspunkt kann Websphere bezüglich der angebotenen Administrationstools verbuchen, die eine verteilte und weitgehenden Verwaltung der Komponenten des Application-Servers zulassen. Apache Tomcat behandelt diesen Punkt nur rudimentär und zwingt den Nutzer zu direkten Eingriffen in die Konfigurationsdateien.

[16] siehe Abschnitt 2.3

Die Entwurfsansätze zu dem Einsatz eines Data Warehouses im Rahmen der Beschaffungsanwendung und darüber hinaus sind als erste Näherungen zu verstehen. Nichtsdestotrotz ist eine Implementierung auf Basis der vom Microsoft SQL Server bereitgestellten Features eine sehr interessante Option für zukünftige Entwicklungen.

6.2 Ausblick

Insbesondere das den vorhergehenden Absatz abschließende Data Warehouse scheint großes Potential für den Einsatz in der Beschaffungsanwendung zu haben. Es kann einerseits zu einer Entlastung der datenbankintensiven Listauswertungen im OLTP-System der Produktionsumgebung führen. Andererseits kann es auch den Rahmen für die Integration von heterogenen Daten darstellen, die beispielsweise aus von der Firma HIS GmbH entwickelten Softwarekomponenten stammen. Dem Controlling der Universitäten könnte so ein mächtiges Werkzeug zur Verwaltungssteuerung an die Hand gegeben werden.

Konkretere Erweiterungen sind die batchgesteuerte Kopplung an das Buchungssystem der Universität (Modul MBS der Firma HIS GmbH) im 4. Quartal 2002. Bestellungen können ab diesem Zeitpunkt von der Einrichtungen manuell und automatisiert auf einen Status gesetzt werden, der die betroffene Bestellung dem Buchungssystem zu festgelegten Zeitpunkten übermittelt. Das Dezernat 1 sowie die Einrichtungen ersparen sich so die manuelle Durchführung von über 4.000 Buchungen jährlich.

Sollten sich für die Beschaffunganwendung Einsatzgebiete außerhalb der Universität Leipzig ergeben, könnten sich aus den vergleichenden Untersuchungen zu den Application-Servern konkrete Änderungen bezüglich der eingesetzten Plattform ergeben. Derzeit wird die Anwendung in einer Produktivumgebung mit dem Application-Server Apache Tomcat betrieben. Aus den vorstehend zusammengefassten Gründen der Zuverlässigkeit und Administrierbarkeit, insbesondere bei mehrfachen Installationen, ist der Einsatz eines stärker optimierten Produkts für den Fall ratsam, dass die Belastung des Systems durch mehrfache Installationen vervielfacht wird.

Literaturverzeichnis

AR99 Aris, A.: *Effizienzsteigerung durch teamorientierte Einkaufskoordination*, in: McKinsey Akzente 10 (1999), S. 8-15. Januar 1999.

AT02 Sun Microsystems: Apache Tomcat – Artikel der Firma Sun Microsystems. http://java.sun.com/products/jsp/tomcat/. 17.08.2002.

BA99 Baker, H. et al.: *E-Sourcing – 21st Century Purchasing*, New York et al. 2000.

BN96 Brandenburger, A.M./Nalebuff, B.J.: *Co-opetition*, New York et al., S. 17. 1996.

BR01 Brosius, G.: *Data Warehouse und OLAP mit Microsoft.* Bonn, Galileo Press. 2001.

EC02 Endrei, M.; Cluning, R.; Daomanee, W.; Heyward, J.; Iyengar, A.; Mauny, I.;Naumann, T.; Sanchez, A.: *IBM Websphere V4.0 Advanced Edition Handbook.IBM*. http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/ sg246176.html?Open. 2002.

ES01 Esan, B.: *Architektur und Arbeitsweise der E-Plattform WebSphere*, Universität Stuttgart. http://www.informatik.uni-stuttgart.de/ipvr/as/lehre/hauptseminar/docws00/WAS-Ausarbeitung.pdf. 2001.

EY99 Ernst&Young: *Content Management – The Critical Success Factor for E-Procurement.* 1999.

FE99 Fehr, B.: *Pioniere im Netz*, Manager Magazin Heft 10, S. 274-283. 1999.

FL02 Flashline Inc.: *Applicationserver-Matrix*. http://www.flashline.com/Components/ appservermatrix.jsp. 2002.

GC00 Gemini Consulting: *Electronic Procurement*, Vorlesung am Lehrstuhl für E-Commerce der Universität Frankfurt. Januar 2000.

GH00 Gunther, H.: *Websphere Application Server Development Best Practices für Performance and Scalability*, IBM White paper. http://www-3.ibm.com/software/webservers/appserv /appserv/ws_bestpractices.pdf. , 2000.

HI02 HIS Hochschul-Informations-System GmbH*: Allgemeine Informationen.* http://www.his.de/doku/his/allgem.htm

JH02 Jungmann, M.; Herbers, J.: *Loadtesting von Web-Applikationen*, DaimlerChrysler Forschung und Technologie, Software-Architektur. http://www.jfs2001.de/folien/E5.pdf. 2002

JE93 Jenz, D.E.: *Datenbanken in Netzwerken – die Komplexität beherrschen*. Handbuch der modernen Datenverarbeitung, Heft 174, Forkel-Verlag, 42-57, 1993.

JT02 Homepage des Apache Jakarta Projekts : http://jakarta.apache.org. 2002.

MÜ01 Müller, Heiko: *Ein Gerücht*. http://www.vocager.de/voold/pages/rubrik1/page1.html, Hauptseite: visual-objects.de, Deutscher Visual Objects Support und Vertrieb. 2002.

NC02 CMP-WEKA GmbH&Co. KG: *Mehr Hubraum für Web-LKWs*, Artikel in www.networkcomputing.de. http://www.networkcomputing.de/heft/solutions/sl-2002/sl_0502_48.htm. 2002.

ND02 CMP-WEKA GmbH&Co. KG: *Motor für Unternehmensherzen*, Artikel in www.networkcomputing.de. http://www.networkcomputing.de/heft/buyers%20guide/bg_0402.html. 2002.

NG99 Nenninger, M./Gerst, M.H.: *Wettbewerbsvroteile durch Electronic Procurement – Strategien, Konzeption und Realisierung*, in: Hermanns, A./Sauter, M. (Hrsg., 1999), S. 283ff. 1999.

PO98a Porter M., *Competitive Strategy*, New York, erste Auflag erschienen 1980. 1998.

PO98b Porter, M., *Competitive Advantage*, New York, erste Auflage erschienen 1985. 1998.

RA94 Rahm, E.: *Mehrrechner-Datenbanksysteme: Grundlagen der verteilten und parallelen Datenbankverarbeitung*. Bonn, Addison-Wesley 1994.

SA02 Sun Microsystems: *Distributed Multitiered Applications*. Artikel der Firma Sun Microsystems über die Architektur von J2EE-Applikationen. http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/Overview2.html# 68378. 2002.

SB02 Sun Microsystems: *Simplified Guide to the Java 2 Platform, Enterprise Edition*. http://java.sun.com/j2ee/j2ee_guide.pdf. 2002

SH02 Shachor, G.: *Working with mod_jk*. http://jakarta.apache.org/tomcat/tomcat-3.3-doc/mod_jk-howto.html. 2002.

SI02 Shachor, G.: *Tomcat 3.3 User's Guide*. http://jakarta.apache.org/tomcat/tomcat-3.3-doc/tomcat-ug.html. 2002

SI97 Sieber, P.: *Die Internet-Unterstützung Virtueller Unternehmen*, in: Schreyögg, G., Sydow, J. (Hrsg.; 1997) Managementforschung 7: Gestaltung von Organisationsgrenzen, S. 145-176. 1997.

SN99 Slaight, T./Norman, A.: *E-Procurement – Fastest Return on Your E-Commerce Investment?*, in: Executive Agenda, o. Jg., Issue III, S. 43-51. 1999.

SP00 Sun Microsystems: *JavaServer Pages 1.1 Specifications*. http://java.sun.com/products/jsp/download.html#specs. 2000.

SS02 Sun Microsystems: *Java 2 Platform, Enterprise Edition: Overview*, http://java.sun.com/j2ee/overview.html. 2002.

ST02 Sauer, T.: *.Net fordert das Java-Lager heraus*, in: Computerwoche Extra, Ausgabe 5, S. 22ff. 28. Juni 2002.

SV00 Sun Microsystems: *Java Servlet 2.2 Specifications*. http://java.sun.com/products/servlet/download.html. 2000.

TA99 Taylor, M.R. et al.: *How Electronic Commerce Is Reshaping Industry Structures*, in: Prism, o.Jg., No. 1, S. 53-79. 1999.

TP00 Turau, V.; Pfeiffer, R.: *Java Server Pages*. 1. Auflage, Heidelberg: dpunkt.verlag. 2000

UL02 Ullenboom, C.: *Java ist auch eine Insel.* Galileo Computing. 2002. http://java-tutor.com/javabuch/online.htm

Vv02 Land Sachsen: *Verwaltungsvorschriften zur Haushaltssystematik* , Berlin. 1990

WA00 Wagner, R.: *Promotion zum Thema: Wissensmanagement im Konzern, Systemtheoretische Perspektiven und Implementierungsansätze.* Deutsche Universität-Verlag, Wiesbaden. 2000.

WI96 Inmon, W.H.: *Building the Data Warehouse*, 2nd ed., John Wiley 1996

WH02 Endrei, M. U.a.. *IBM Websphere V4.0 Advanced Edition* Handbook. http://publib-b.boulder.ibm.com/redbooks.nsf/RedbookAbstracts/sg246176.html?Open

Abbildungsverzeichnis

Tabellenverzeichnis

Anhang

A. Die J2EE-Spezifikation

Die J2EE wurde für den Bereich unternehmensweiter Anwendungen konzipiert. Sie besteht aus insgesamt vier Teilen:

- Design- und Programmiermodell,
- J2EE-Plattform,
- Test-Suite,
- Referenzimplementierung

Die folgenden Absätze behandeln detaillierter als es in dem Hauptteil der Arbeit sinnvoll war, um einen vollständigeren Einblick in die Architektur zu geben.

A.1 J2EE – Design- und Programmiermodell

Dieses Modell stellt ein Konzept für eine komponentenbasierte, modulare, schichten-orientierte und verteilte Softwareentwicklung dar, welches aus mehreren klaren, logischen Schichten besteht. Die J2EE-Applikationen sind jeweils nach ihrer Funktion in mehrere Komponenten aufgeteilt. Diese sind auf verschiedenen Rechner-Systemen installiert, abhängig von der Schicht, zu der die jeweilige Applikationskomponente gehört. Obwohl J2EE-Applikationen über drei, vier oder mehr Schichten verteilt sein können, wird generell von einer *Drei-Schicht-Architektur* ausgegangen, die in der folgenden Abbildung schematisch dargestellt wird:
Für die Darstellung weitergehender Informationen zum J2EE – Design–und Programmiermodell sei an dieser Stelle auf den Abschnitt 1.3.2 verwiesen, der diesen Bereich zur Einordnung der entwickelten Anwendung detailliert behandelt.

A.2 J2EE – Plattform

Die J2EE-Plattform definiert eine Reihe von APIs sowie auf XML basierende Konfigurationstechniken die für die Implementierung von mehrschichtigen J2EE-Applikationen erforderlich sind.
Alle Produkte wie bspw. Application-Server, die zu J2EE kompatibel sein wollen, müssen diese Standards unterstützen.

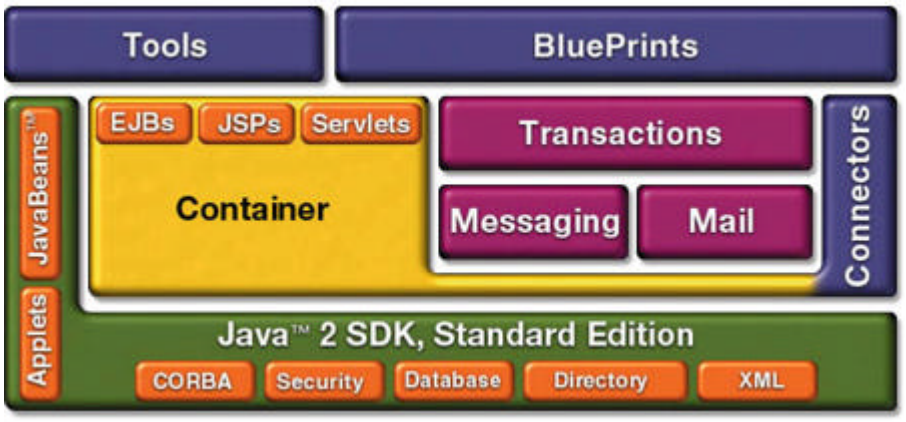


Abbildung 0-1: Übersicht der J2EE-APIs [SS02]

Dies stellt sicher, dass die jeweilige Plattform eine konsistente Umgebung für die Entwicklung von J2EE-Applikationen darstellt. Andererseits müssen Applikationen bedingt durch den Aufbau der J2EE-Plattform eine Reihe von Bestandteilen enthalten.
Im folgenden werden erst die J2EE-APIs genannt, die eine Plattform zur Verfügung stellen muss und anschließend die Bestandteile einer J2EE-Applikation.

J2EE-APIs

JDBC (Java DataBase Connectivity)

Diese API realisiert den Zugriff auf relationale Datenbankmanagement-Systeme. Es bietet u.a. eine Unterstützung von SQL3-Datentypen, verteilte Transaktionen und Connection Pooling.

JTA/JTS (Java Transaction API / Java Transaction Service)

JTA definiert eine Reihe von Schnittstellen, welche die Kommunikation mit einem Transaktionsmanager ermöglichen.

JTS stellt einen Transaktionsmanager, basierend auf der Spezifikation des Transaktionsdienstes der OMG, dar. Hierzu integriert der JTS einen Transaktionssteuerungsdienst, auf die der Server zurückgreifen kann.

JMS (Java Message Service)

Der Java Message Service ist eine API für einen gesicherten Punkt-zu-Punkt Datentransfer, wofür ein JMS-Anbieter benötigt wird. Mittels JMS ist der Application Server in der Lage, Daten asynchron zu versenden und zu empfangen.

JNDI (Java Naming and Directory Interface)

Die JNDI-API bietet Zugriff auf Namens- und Verzeichnisdienste, u.a. LDAP oder das Novell Directory Interface.

JavaMail

Hiermit können aus J2EE-Applikationen heraus E-Mails generiert und versendet werden.

BluePrints

Die Java BluePrints definiert ein Programmiermodell für J2EE-Applikationen, indem den Entwicklern Richtlinien, Musterapplikationen, diverse Tools und freierhältlicher Quellcode von Demo-Programmen angeboten werden.

Bestandteile einer J2EE-Applikation

Die Bestandteile von J2EE-Applikationen können in drei Kategorien eingeteilt werden:

Komponenten

Die J2EE-Spezifikation definiert zwei Arten von Server-Komponenten:

- Web-Komponenten: Sie generieren die Benutzeroberfläche und steuern die Interaktion mit dem Client in einer web-basierten Anwendung. Hierzu gehören *Java Servlets* und J*avaServer Pages.*
- Enterprise Java Beans (EJB): Die EJB-API ist eine Spezifikation für verteilte Java Beans auf der Basis von RMI (Remote Method Invocation) für die Implementierung von Business-Logik. [TP00, Seite 17]

Container

Sie bilden als Schnittstelle zu plattformspezifischen Funktionalitäten die Laufzeitumgebung für die Komponenten. Findet ein Applikations-Client als Frontend Anwendung, hat der *Applikations-Client-Container* dessen Ausführung als Aufgabe. Sogenannte *Servlet-Container* dienen zur Ausführung von Web-Komponenten. Ein *JSP-Container* besitzt die Aufgabe, eine JSP-Seite in ein Servlet zu konvertieren, welches dann durch den Servlet-Container ausgeführt wird. In Analogie dazu führen *EJB-Container* Enterprise Java Beans aus.

Abbildung 0-2 gibt eine Übersicht über die Orte der verschiedenen Container-Typen im Drei-Schicht-Architekturmodell.

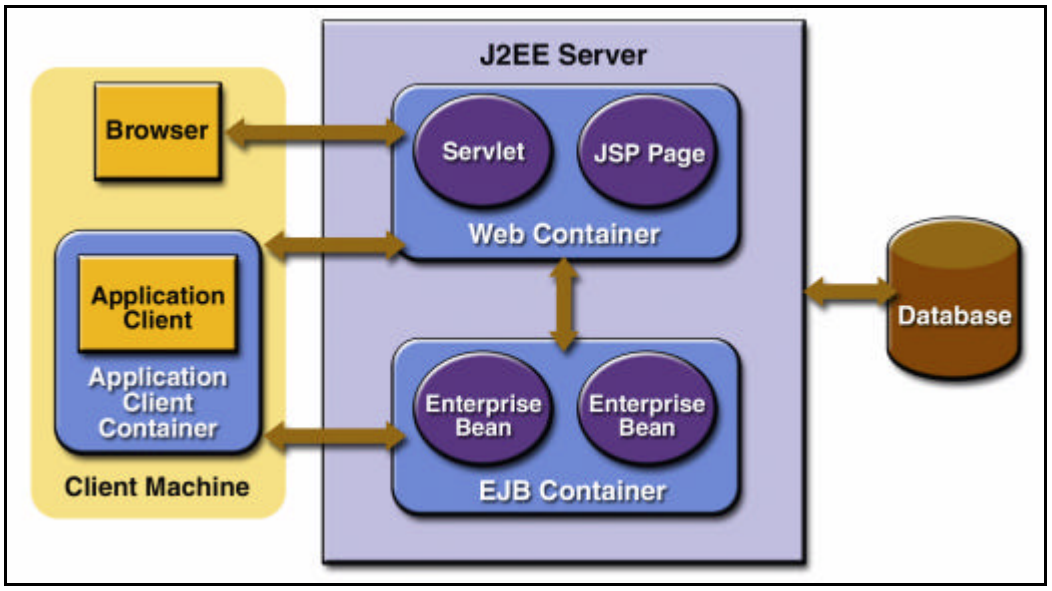


Abbildung 0-2: Containertypen in der Drei-Schicht-Architektur [SS02]

Connectoren

Sie besitzen die Aufgabe, eine Standard-Architektur für die Anbindung der J2EE-Plattform an heterogene Unternehmensinformationssysteme (EIS) wie Transaktionsmonitore oder Datenbanken zu definieren. In dieser Architektur sind Sicherheits- und Transaktionsmechanismen enthalten, welche eine Anbindung von Unternehmensinformations-Systemen an Application Server ermöglichen.

Die J2EE-Connector Architektur ermöglicht EIS-Anbietern die Herstellung eines sogenannten *Standard Resource Adapters*, welcher in einen Application Server integriert wird. Er schafft die Verbindung zwischen dem EIS und dem Application Server. Der Standard Resource Adapter wird in jedem Application Server lauffähig sein, der die Connector-Architektur unterstützt. Dazu muss auch der Anbieter des jeweiligen Application Servers die Unterstützung der J2EE-Connector Architektur gewährleisten. [TP00, Seite 17]

A.3 J2EE - Test Suite

Hierzu gehören Programme, mit denen die Kompatiblität von Implementationen mit dem J2EE-Standard verifiziert werden kann.

Die J2EE Test Suite beinhaltet Tests für alle Klassen und Methoden, welche von der J2EE-Spezifikation benötigt werden. Darin sind ebenfalls Tests enthalten, welche die schichtenübergreifenden Interaktionen einer J2EE-Applikation auf Korrektheit und Konsistenz prüfen.

A.4 J2EE – Referenzimplementierung

Die J2EE-Spezifikation enthält auch die Referenzimplementierung eines Application- Servers. Seine Hauptaufgabe liegt in der Demonstration der spezifizierten Technologien des J2EE-Standards. Hiermit haben Anbieter bei der Entwicklung ihrer eigenen Implementierungen die Möglichkeit, sie auf Basis eines sich exakt an die Spezifikationen haltenden Application-Servers auf Kompatibilität zu testen.
Eine weitere Funktion der Referenzimplementierung ist es, eine Standardplattform zu definieren, welche als Basis für die J2EE Test Suite dient [SB02].
Des weiteren ist es ein frei erhältliches Produkt, dessen Quellcode offen liegt und so zur Popularisierung der Java 2 Plattform, Enterprise Edition beiträgt. Somit kann es zu Demonstrationszwecken und akademischen Forschungen dienen.

A.4.1 Java Servlets und JavaServer Pages

Java Servlets und JavaServer Pages sind Bestandteile der J2EE von Sun Microsystems und dienen zum Erstellen serverseitig auszuführender Applikationen. Beide Technologien basieren auf der Programmiersprache Java, wobei Servlets einen prozeduralen und JavaServer Pages einen deklarativen Ansatz verfolgen.
Bedingt durch den prozeduralen Stil sind Servlets eigenständige Java-Programme, die, ebenso wie JavaServer Pages, bei jeder Client-Anfrage reine HTML- oder XML-Seiten zurückliefern. In Servlets ist jedoch das Design in der jeweiligen Auszeichnungssprache und die Applikationslogik untrennbar miteinander verbunden, was bei der Erstellung komplexer Webseiten problematisch ist.

> Java Servlet Spezifikation:
> *A servlet is a Java technology based web component, managed by a container, that generates dynamic content. Like other Java-based components, servlets are platform independent Java classes that are compiled to platform neutral bytecode that can be loaded dynamically into and run by a Java enabled web server.*[SC02]
>
> JavaServer Pages Spezifikation:
> *JavaServer Pages™ is the Java™ 2 Platform, Enterprise Edition (J2EE) technology for building applications for generating dynamic web content, such as HTML, DHTML, XHTML and XML.* [SD02]

Der deklarative Stil der JSP-Technologie ermöglicht eine bessere Trennung von Design und Anwendungslogik bei der Generierung eines Dokumentes. Dies wird durch die enge Integration von JavaBeans, dem Softwarekomponentenmodell von Java, und durch benutzerdefinierte Tag-Bibliotheken erreicht.

JavaBeans sind wiederverwendbare Softwarekomponenten, welche die JavaBeans-Spezifikation erfüllen müssen. Sie können in JSP-Seiten dazu benutzt werden, Webapplikationen besser zu strukturieren und den Anteil von Java-Code möglichst gering zu halten.

A.4.2 Die Ausführung einer JSP-Seite

Da eine JSP-Seite nicht wie eine HTML-Seite zum Client geschickt werden kann, weil sie sowohl aus HTML- als auch aus Java-Code besteht und somit von einem Webbrowser nicht interpretiert werden kann, wird eine angeforderte JSP-Datei auf dem J2EE-Server zunächst in den Quellcode eines Servlets umgewandelt und anschließend kompiliert. Dies bezeichnet man als Übesetzungsphase und wird nur ausgeführt, wenn die JSP-Seite auf dem Server installiert wird oder die erste Anfrage erfolgt.
Bei jeder Anfrage an diese Seite wird das zuvor erzeugte und übersetzte Servlet ausgeführt. Vor der Ausführung des Servlets überprüft der JSP-Container, ob die JSP-Seite seit dem letzten Übersetzungsvorgang modifiziert wurde. In diesem Fall wird ein neues Servlet erzeugt, kompiliert und ausgeführt. Bei der Ausführung dieses Servlets wird eine entsprechende HTML-Seite generiert und als Antwort an den Client gesendet.

A.4.3 Vergleich von Servlets und JavaServer Pages mit ähnlichen Technologien

Neben der Java Servlets- und JavaServer Pages-Technologie der Firma Sun Microsystems existieren noch weitere serverseitige Technologien für die Erstellung von dynamischen Webseiten. Die früheste und Entwicklung dieser Art war das Common Gateway Interface (CGI). Daneben haben Hersteller von Webservern eigene Schnittstellen geschaffen, wie zum Beispiel das ISAPI von Microsoft sowie das NSAPI von Netscape.
Alle genannten Schnittstellen befähigen den Webserver, externe Programme aufzurufen, welche anschließend eine HTML-Seite generieren und diese zum Client zurückschicken.

Common Gateway Interface

Das Common Gateway Interface wurde vom NCSA (National Center for Supercomputing Applications) definiert als erster Standard zur Spezifikation der Verbindung von Webservern und externen Programmen. Es spezifiziert einen Mechanismus für Webserver, mit dem sie Parameter an externe Programme übergeben und diese aufrufen können.
Obwohl die Programmiersprache PERL die wahrscheinlich größte Verbreitung in diesem Ansatz gefunden hat, können prinzipiell Programme jeglicher Programmiersprache, die auf dem lokalen Rechner installiert ist, per CGI ausgeführt werden, so bspw. auch vorkompilierte Java-Programme.
Der Einsatz von Servlets und JavaServer Pages, also ebenfalls Java-Programmen, verfolgt aber einen anderen Ansatz. Entweder übernimmt der Webserver eine vollständige Einbettung der Java VM oder

wie in den meisten bekannten J2EE-Konfigurationen ein Application Server. Erst innerhalb dieser Java VM werden Servlets und JSPs ausgeführt.
Im folgenden werden die beiden Ansätze anhand einiger herausgegriffener Kriterien verglichen [vgl. UL02, Abschnitt. 18.1.2.].

Performance

Servlets besitzen gegenüber herkömmlichen CGI-Programmen eine wesentlich bessere Performance, da die Java VM direkt in den Web- oder Application Server integriert ist und damit nicht externe Programme gestartet werden müssen. Dies ist bei der CGI-Technologie ein Nachteil, denn jede Client-Anfrage startet ein externes Programm und damit einen externen Prozess. Bei Servlets dagegen wird jede Anfrage durch ein seperates Thread-Objekt gehandhabt, welche in den Punkten Speicherverbrauch und Erzeugung wesentlich optimaler als externe Programme sind.

Einfache Beschaffenheit und Portabilität

Mit der Möglichkeit der uneingeschränkten Nutzung von Java steht dem Programmierer eine weit verbreitete und leistungsfähige Programmiersprache zur Verfügung. Insbesondere durch die Nutzung von standardisierten Klassenbibliotheken, den Java-APIs, und der objektorientierten Eigenschaften der Sprache kann eine einfache Wartung und Erweiterung der Software erreicht werden. Bedingt durch diese Tatsache, sind Java Servlets und JavaServer Pages ebenfalls plattformunabhängig. Durch ihre Verfügbarkeit auf allen gängigen Plattformen ist ein Austausch zwischen verschiedenen Plattformen und Webservern meist mit minimalem Portierungsaufwand möglich.

Datenaustausch und Kommunikation mit dem Webserver

Da ein CGI-Programm unabhängig von anderen läuft und eigene Zustände verwaltet, gestaltet sich die Kommunikation und Datenteilung mit weiteren CGI-Programmen als schwierig. Java-Servlets dagegen laufen in einem gemeinsamen Maschinenkontext und können daher leicht untereinander Daten austauschen. So können beispielsweise aus Optimierungsgründen Datenbankverbindungen gemeinsam genutzt werden.

Unterstützung von Sessions

Servlets und JavaServer Pages besitzen im Gegensatz zur CGI-Technologie eine standardmäßige Unterstützung von Sessions. Eine Session stellt eine zusammenhängende Abfolge von HTTP-Anfragen und -Antworten bei einer Client-Server-Kommunikation dar. Die Implementierung von Sessions ist notwendig geworden, da das HTTP ein zustandsloses Protokoll ist. Aus diesem Grund muss bei jeder Anfrage und Antwort eine eindeutige

Identifikation mitgesendet werden, damit es für den Server möglich ist, mehrere Anfragen desselben Clients als zusammenhängend erkennen zu können.

Active Server Pages

Microsoft Active Server Pages (ASP) stellt eine weitere serverseitige Technologie dar, welche JavaServer Pages ähneln und ebenfalls in HTML-Seiten eingebetteten Programmcode für die Erzeugung dynamischer Webseiten enthält. Sie besteht aus einem Skriptinterpreter, welcher in den Internet Information Server, einem Webserver der Firma Microsoft, integriert ist. Bedingt durch diese Architektur bestehen gegenüber JavaServer Pages einige Nachteile:

- In ASP können nur die Skriptsprachen VisualBasicScript und JScript von Microsoft verwendet werden, die bei komplexen Vorhaben an Grenzen stoßen.
- JavaServer Pages und Servlets befinden sich nach ihrer ersten Ausführung im Speicher der Java VM, so dass bei jedem weiteren Aufruf ein schneller und effizienter Zugriff erfolgen kann. Active Server Pages dagegen werden vom Interpreter bei jeder Ausführung neu aufgerufen.
- Im Gegensatz zu JavaServer Pages sind Active Server Pages nicht webserver- und plattformunabhängig, da als Webserver nur der IIS eingesetzt werden kann. Dieser wiederum ist ausschließlich auf Windows-Betriebssystemen lauffähig.

Hypertext Preprocessor

PHP ist eine Open-Source-Skriptsprache, welche in HTML-Seiten eingebettet werden kann. Auch hier gibt es also Analogien im Konzept der Technologie zu ASP und JSP. Nachteilig wirkt sich aus, dass keine bereits bestehenden Sprachen wie Perl, Visual Basic, C oder Java eingebunden werden können und daher eine komplett neue Skriptsprache erlernt werden muss. JavaServer Pages bieten den Vorteil, dass mit dem Einsatz von Java auf eine Vielzahl bestehender APIs zurückgegriffen werden kann.

B. Weitere Funktionalitäten von Application-Servern

Neben der bisher genannten Basisfunktionalität, wie z.B. der Beherbergung der Hauptlogik der Unternehmensanwendung in Form von J2EE-Komponenten innerhalb der entsprechenden Container und die Anbindung an verschiedene Backendsysteme, besitzen javabasierte Application-Server zahlreiche weitere typische Funktionen und Eigenschaften, um für unternehmenskritische Anwendungen ein Höchstmaß an Ausfallsicherheit und Skalierbarkeit zu erreichen.
Dazu gehören vor allem:

Load-Balancing

Unter *Load-Balancing* versteht man die Verteilung von Anfragen, die einen Server erreichen, auf mehrere dahinterliegende Server, die diese Anfragen dann parallel abarbeiten. Der Server, der für die Verteilung der Anfragen verantwortlich ist, wird als Load-Balancer bezeichnet.

In der ersten Stufe ist Load-Balancing zwischen Webservern und Web-Containern möglich. Dazu verteilen ein oder mehrere Webserver die Servlet- und JSP-Aufrufe an die dahinter liegenden Application Server. Dies kann was beispielsweise über das Apache-Jserv-Protocol (AJP) erfolgen. Besitzen die Application Server die Fähigkeit, Web- und EJB-Container auf seperate Server zu verteilen, dann können Web-Container in der zweiten Stufe EJB-Aufrufe beispielsweise per JNDI an die dahinter liegenden EJB-Container verteilen. Diese Trennung von Webserver, Web- und EJB-Container ist besonders bei der Auslieferung vieler statischer Webseiten vorteilhaft. Weiterhin ermöglicht dieser Ansatz eine gleichmäßigere Verteilung der unterschiedlichen Lasten von Web- und EJB-Ebene.

Fail-Over

Unter *Fail-Over* versteht man eine Technik zur Erhöhung der Verfügbarkeit von J2EE-Anwendungen, indem mit der es für andere Web- oder EJB-Container möglich ist, die Daten und Aufgaben eines ausfallenden Application-Servers zu übernehmen. Sogenanntes *Session-Fail-Over* sorgt dafür auf Session-Ebene. Dazu werden die Server zu einem Cluster zusammengefasst und und die Sessioninformationen über alle Server, die sich in diesem Cluster befinden, repliziert. Bei einem Ausfall eines EJB-Containers dagegen lesen die EJBs die entsprechenden Informationen entweder aus Dateien oder einer Datenbank aus. [NC02]

Connection-Pooling

Application Server besitzen meist auch das sogenannte *Connection-Pooling* zur Steigerung der Performance. Darunter versteht man die Fähigkeit, einen Pool von Verbindungen zu einer oder mehreren Datenbanken offen zu halten. Bei einer Datenbankabfrage benutzt der Application Server eine bereits bestehende Verbindung aus diesem Pool, um einen zeitintensiven Verbindungsauf- und Abbau zu einer Datenbank zu vermeiden. [ND02]

Caching

Das *Caching* ist eine weitere Technik von Application Servern zur Performanceerhöhung. Das sogenannte *Result-Caching* dient, ebenso wie das Connection-Pooling, zur Beschleunigung von Datenbankabfragen. Hierzu werden Parameter und Ergebnisse einer bereits gemachten Datenbankabfrage in einem Cache zwischengespeichert. Bei einer weiteren Datenbankanfrage prüft der Application Server, ob die Ergebnisse bereits im Cache vorhanden sind. Ist dies nicht der Fall, wendet er sich direkt an die Datenbank.

Aber auch an anderen Stellen einer J2EE-Applikation kann durch Caching die Anwortzeit zu einem Client verringert werden. Einige Application Server sind in der Lage, statische und dynamische Web-Komponenten in sogenannten Cache-Servern zwischenzuspeichern. Dabei ist die Caching-Technologie umso effektiver, je häufiger die Abfrage identischer Daten von Anwendern oder Programmkomponenten ausgelöst wird. [ND02]

Sicherheitsmechanismen

Zur sicheren Übertragung von Daten zwischen J2EE-Client und Application Server kann der Application-Server die SSL-Technologie und das HTTPS-Protokoll implementieren. Dazu werden zur gegenseitigen Identifizierung Zertifikate benutzt. Der JAAS (Java Authentification And Authorization Service) implementiert für Java zur Autorisierung und Authentifizierung das PAM-Framework (Pluggabe Authentification Module). [ND02]

Session-Management

Resultierend aus der Tatsache, dass das HTTP ein zustandsloses Protokoll ist, kann ein Server nicht feststellen, ob eine Folge von HTTP-Anfragen von demselben Client stammt. Für einfache Web-Applikationen ist dies in der Regel ausreichend. Moderne Anwendungen benötigen jedoch ein Sitzungskonzept, damit der Server mehrere Anfragen einem Client eindeutig zuordnen und Informationen über Anfragen hinweg verwalten kann.

Damit ein Server erkennen kann, ob eine HTTP-Anfrage zu ein und derselben Sitzung gehört, muss der Client bei jeder Anfrage eine Information über sich zur eindeutigen Identifikation mitsenden. Sie wird im folgenden Session-ID genannt.

Die J2EE-Spezifikationen erlauben das Ablegen beliebiger Objekte in Sessions. Dies ermöglicht die Zwischenspeicherung komplexer Datenstrukturen sowie großer Datenmengen. Weil sich die Nutzung von Sessions in der Implementierung sehr einfach gestaltet, sind sie ein gern eingesetztes Hilfsmittel.

Es gibt verschiedene Möglichkeiten, wie die Übertragung der Session-ID zwischen Server und Client erfolgen kann. Die wichtigsten davon sind:

- Cookies

  Als *Cookie* bezeichnet man kleine Informationseinheiten, welche Clients, meist in Form eines Webbrowsers, und Webserver untereinander austauschen. Dazu verschickt der Server mit dem Header der HTTP-Antwort beliebig viele Cookies. Diese werden vom Client in Form von Dateien gespeichert und bei jedem weiteren Aufruf einer Seite dieses Servers mitgesendet. Obwohl Cookies bisher nicht in die HTTP-Spezifikation aufgenommen wurde, ist heute eine Unterstützung von fast allem Browsern vorhanden.

Nachteil dieses Verfahrens ist, dass Cookies vom Client deaktiviert werden können, sodass kein Session-Management mehr betrieben werden kann.

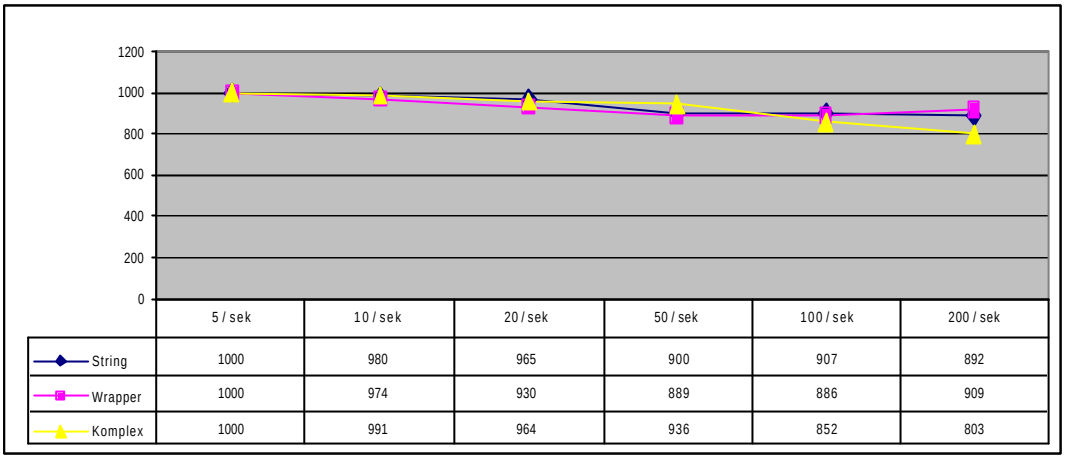
- Umschreiben von URLs (URL-Rewriting)

  Eine weitere Möglichkeit besteht darin, eine URL dynamisch zu modifizieren, indem ein Wert angehängt wird, welcher die aktuelle Session kennzeichnet. Dieser Wert enspricht dann genau dem des Cookies. Vorteilhaft daran ist, dass dieses Verfahren mit allen Browsern funktioniert, selbst wenn Cookies explizit abgelehnt werden. Nachteilig an diesem Verfahren ist, dass jede URL modifiziert werden muss, weshalb eine dynamische Generierung der Webseiten erforderlich ist. Dies bedeutet einen höheren serverseitigen Aufwand bei der Erstellung der Servlets und JSP-Seiten. Ein weiteres schwerwiegendes Problem ist, dass diese dynamisch erzeugten URLs in Bookmarks der Webbrowser übernommen werden können, so dass später Sessions angesprochen werden, die gar nicht mehr existieren können. Um die Wahrscheinlichkeit eines Missbrauch zu reduzieren, wird die Gültigkeitsdauer von Sessions so kurz wie möglich gesetzt und der Nutzer aufgefordert, sich nach der Nutzung einer Applikation explizit auszuloggen.

- SSL-Sessions

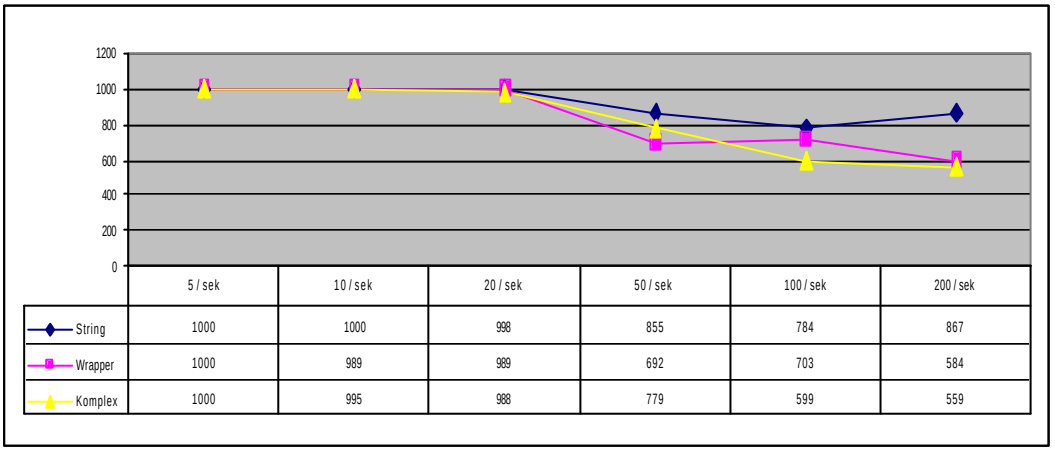
  Der *Secure Socket Layer*, der für die verschlüsselte Datenübertragung im HTTPS-Protokoll verantwortlich ist, besitzt ein explizites Session-Konzept, welches ebenfalls zur Betreibung des Session-Managements genutzt werden kann. Vorteilhaft an dieser Technologie ist, daß keine spezielle Software auf der Clientseite für die Verschlüsselung der Daten erforderlich ist. Allerdings ist eine Implementierung des HTTPS-Protokolls in den Webserver notwendig. Eine Weiterentwicklung des SSL stellt das *Transport-Layer-Security*-Protokoll dar, welches ebenfalls ein Session-Konzept enthält. [TP00, Seite 200]

C. Messergebnisse zum Session-Managements im Detail



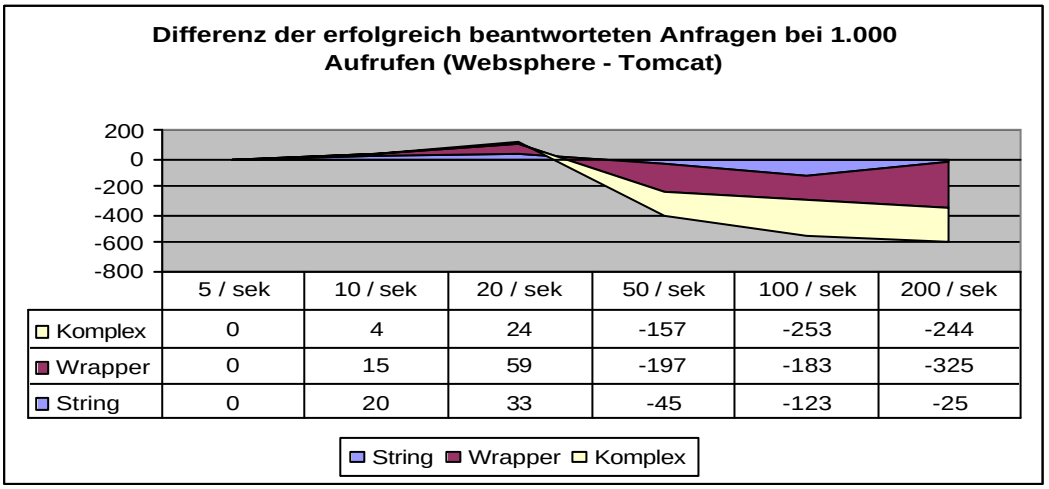
	5 / sek	10 / sek	20 / sek	50 / sek	100 / sek	200 / sek
String	1000	980	965	900	907	892
Wrapper	1000	974	930	889	886	909
Komplex	1000	991	964	936	852	803

Abbildung 0-3: Durchsatz von Tomcat bei 1.000 Anfragen (X-Achse logarithmisch)



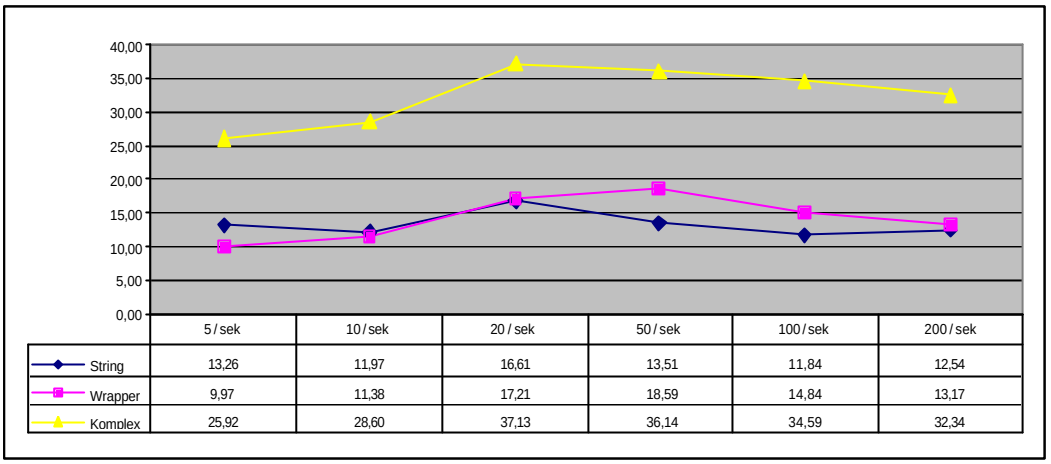
	5 / sek	10 / sek	20 / sek	50 / sek	100 / sek	200 / sek
String	1000	1000	998	855	784	867
Wrapper	1000	989	989	692	703	584
Komplex	1000	995	988	779	599	559

Abbildung 0-4: Durchsatz von Websphere bei 1.000 Anfragen (X-Achse logarithmisch)



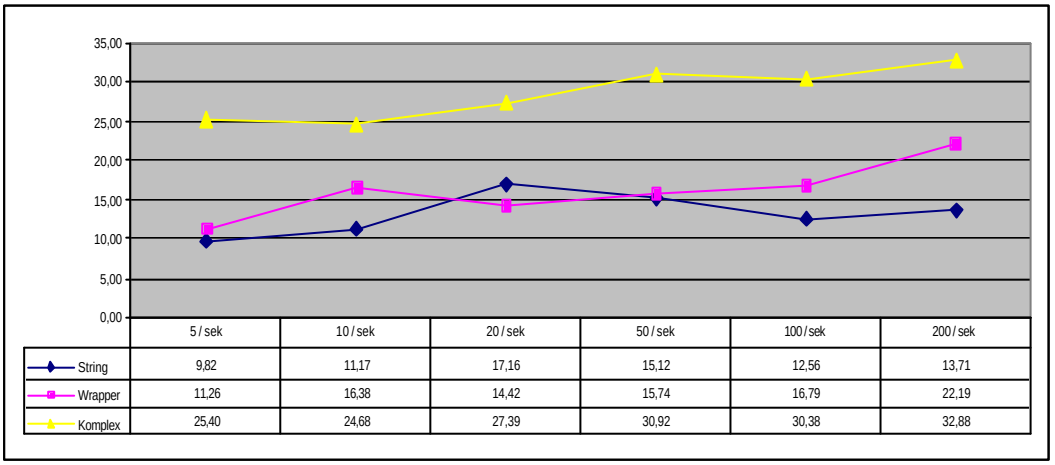
	5 / sek	10 / sek	20 / sek	50 / sek	100 / sek	200 / sek
Komplex	0	4	24	-157	-253	-244
Wrapper	0	15	59	-197	-183	-325
String	0	20	33	-45	-123	-25

Abbildung 0-5: Differenz der erfolgreich beantworteten Anfragen bei 1.000 Aufrufen (Websphere - Tomcat) (X-Achse logarithmisch)



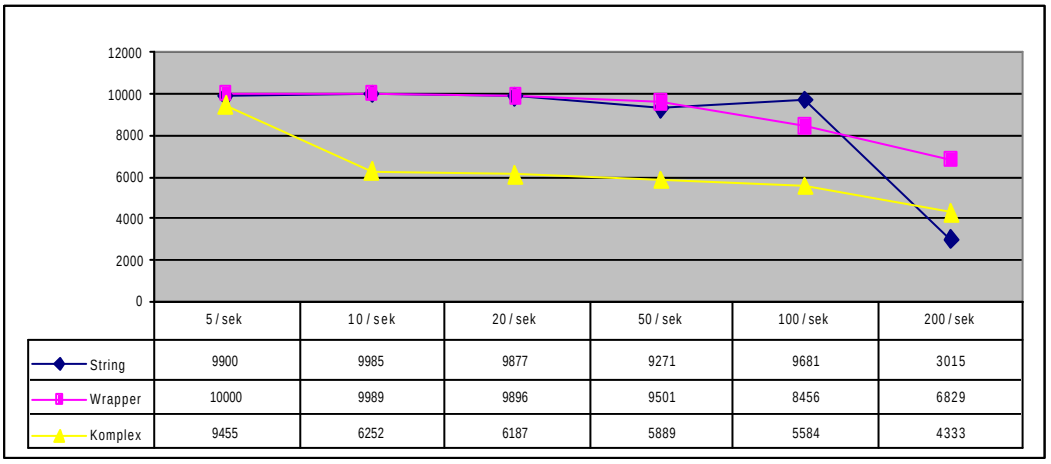
	5 / sek	10 / sek	20 / sek	50 / sek	100 / sek	200 / sek
String	13,26	11,97	16,61	13,51	11,84	12,54
Wrapper	9,97	11,38	17,21	18,59	14,84	13,17
Komplex	25,92	28,60	37,13	36,14	34,59	32,34

Abbildung 0-6: Durchschnittliche Antwortzeiten von Tomcat bei 1.000 Aufrufen in Millisekunden (X-Achse logarithmisch)



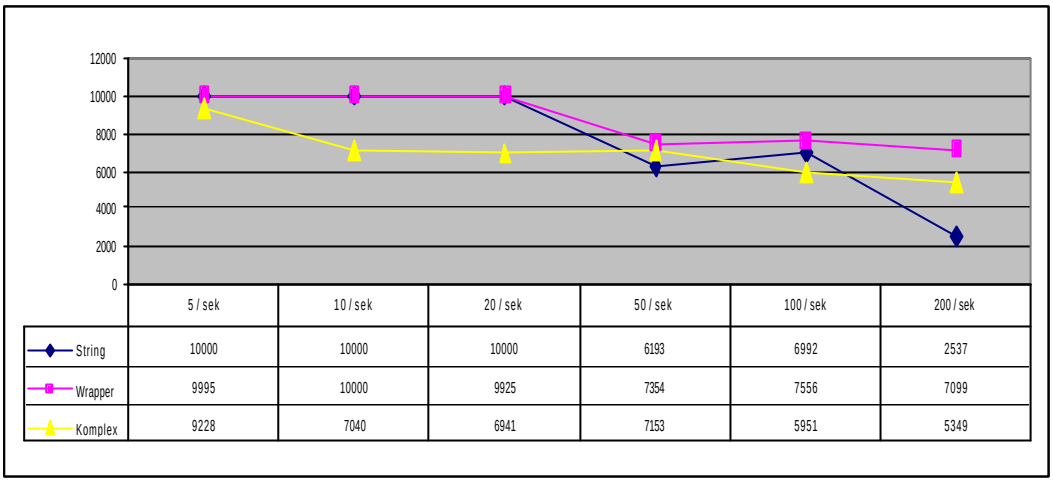
	5 / sek	10 / sek	20 / sek	50 / sek	100 / sek	200 / sek
String	9,82	11,17	17,16	15,12	12,56	13,71
Wrapper	11,26	16,38	14,42	15,74	16,79	22,19
Komplex	25,40	24,68	27,39	30,92	30,38	32,88

Abbildung 0-7: Durchschnittliche Antwortzeiten von Websphere Application Server bei 1.000 Aufrufen in Millisekunden (X-Achse logarithmisch)



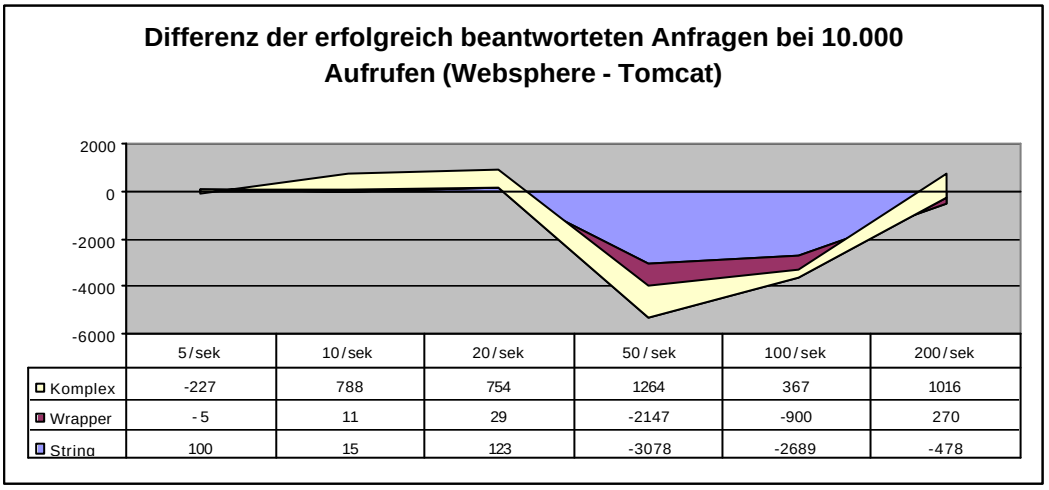
	5 / sek	10 / sek	20 / sek	50 / sek	100 / sek	200 / sek
String	9900	9985	9877	9271	9681	3015
Wrapper	10000	9989	9896	9501	8456	6829
Komplex	9455	6252	6187	5889	5584	4333

Abbildung 0-8: Durchsatz von Tomcat bei 10.000 Anfragen (X-Achse logarithmisch)



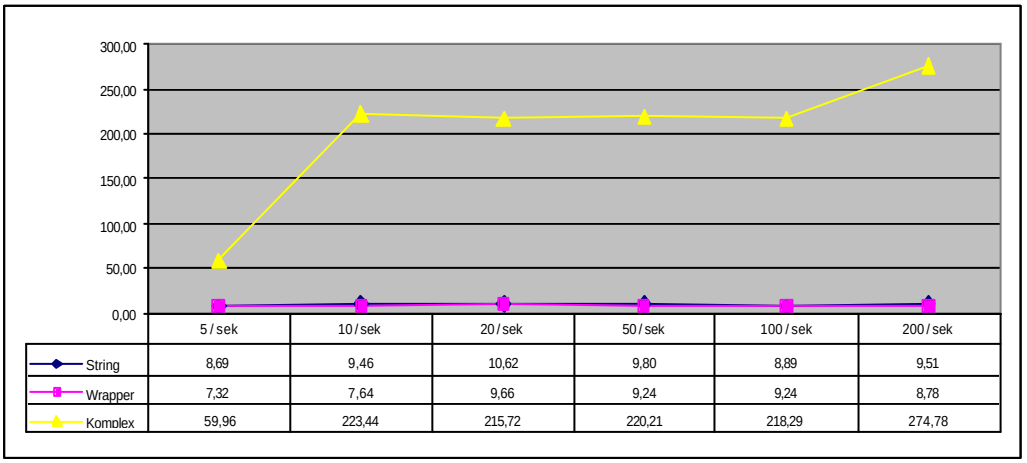
	5 / sek	10 / sek	20 / sek	50 / sek	100 / sek	200 / sek
String	10000	10000	10000	6193	6992	2537
Wrapper	9995	10000	9925	7354	7556	7099
Komplex	9228	7040	6941	7153	5951	5349

Abbildung 0-9: Durchsatz von Websphere bei 10.000 Anfragen (X-Achse logarithmisch)



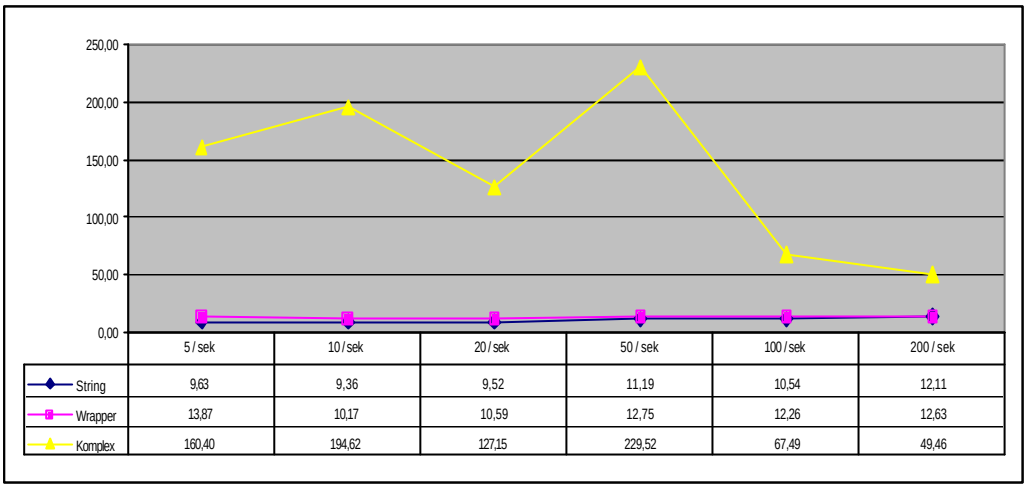
	5 / sek	10 / sek	20 / sek	50 / sek	100 / sek	200 / sek
Komplex	-227	788	754	1264	367	1016
Wrapper	- 5	11	29	-2147	-900	270
String	100	15	123	-3078	-2689	-478

Abbildung 0-10: Differenz der erfolgreich beantworteten Anfragen bei 10.000 Aufrufen (Websphere - Tomcat) (X-Achse logarithmisch)



	5 / sek	10 / sek	20 / sek	50 / sek	100 / sek	200 / sek
String	8,69	9,46	10,62	9,80	8,89	9,51
Wrapper	7,32	7,64	9,66	9,24	9,24	8,78
Komplex	59,96	223,44	215,72	220,21	218,29	274,78

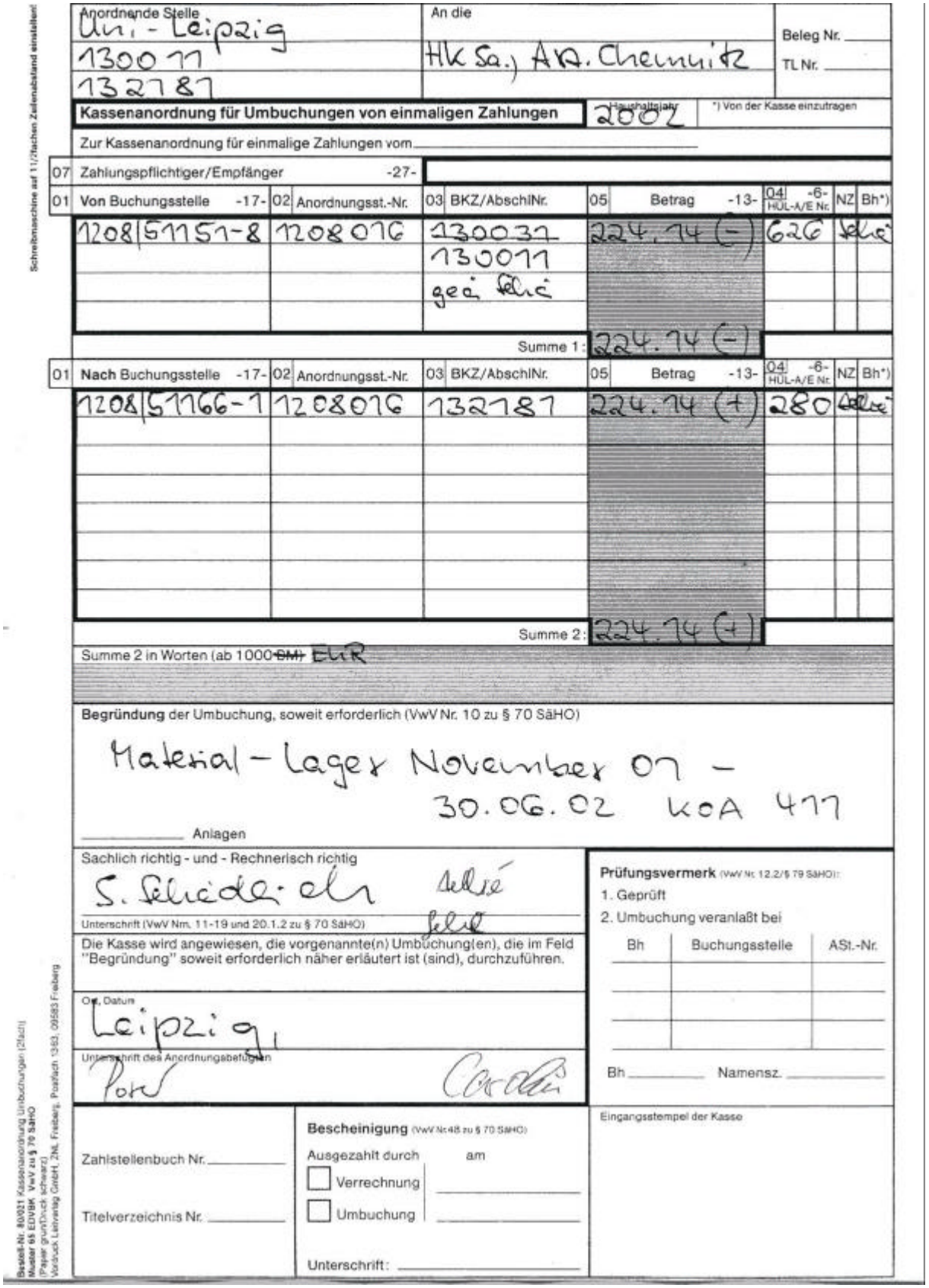
Abbildung 0-11: Durchschnittliche Antwortzeiten von Tomcat bei 10.000 Aufrufen in Millisekunden (X-Achse logarithmisch)



	5 / sek	10 / sek	20 / sek	50 / sek	100 / sek	200 / sek
String	9,63	9,36	9,52	11,19	10,54	12,11
Wrapper	13,87	10,17	10,59	12,75	12,26	12,63
Komplex	160,40	194,62	127,15	229,52	67,49	49,46

Abbildung 0-12: Durchschnittliche Antwortzeiten von Websphere Application Server bei 10.000 Aufrufen in Millisekunden (X-Achse logarithmisch)

D. Ausgefülltes Umbuchungsformular



Anordnende Stelle: Uni. - Leipzig 130077 132787

An die: HK Sa.) AA. Chemnitz

Beleg Nr. ____ TL Nr. ____

Kassenanordnung für Umbuchungen von einmaligen Zahlungen — Haushaltsjahr: 2002 — *) Von der Kasse einzutragen

Zur Kassenanordnung für einmalige Zahlungen vom ____

07 Zahlungspflichtiger/Empfänger -27-

01 Von Buchungsstelle -17-	02 Anordnungsst.-Nr.	03 BKZ/AbschlNr.	05 Betrag -13-	04 -6- HÜL-A/E Nr.	NZ	Bh*)
1208 51151-8	1208076	130037 130077 gez. Schi	224.74 (-)	626		Schi
			Summe 1: 224.74 (-)			

01 Nach Buchungsstelle -17-	02 Anordnungsst.-Nr.	03 BKZ/AbschlNr.	05 Betrag -13-	04 -6- HÜL-A/E Nr.	NZ	Bh*)
1208 51166-1	1208076	132787	224.74 (+)	280		Schi
			Summe 2: 224.74 (+)			

Summe 2 in Worten (ab 1000 ~~DM~~ EUR): ____

Begründung der Umbuchung, soweit erforderlich (VwV Nr. 10 zu § 70 SäHO)

Material - Lager November 07 - 30.06.02 KoA 477

____ Anlagen

Sachlich richtig - und - Rechnerisch richtig

S. Schicke-ehl Schi

Unterschrift (VwV Nrn. 11-19 und 20.1.2 zu § 70 SäHO) Schi

Die Kasse wird angewiesen, die vorgenannte(n) Umbuchung(en), die im Feld "Begründung" soweit erforderlich näher erläutert ist (sind), durchzuführen.

Ort, Datum: Leipzig, 1.

Unterschrift des Anordnungsbefugten: Pohl

Prüfungsvermerk (VwV Nr. 12.2/§ 79 SäHO):

1. Geprüft
2. Umbuchung veranlaßt bei

Bh	Buchungsstelle	ASt.-Nr.

Bh ____ Namensz. ____

Zahlstellenbuch Nr. ____

Titelverzeichnis Nr. ____

Bescheinigung (VwV Nr. 48 zu § 70 SäHO)

Ausgezahlt durch ____ am ____

☐ Verrechnung

☐ Umbuchung

Unterschrift: ____

Eingangsstempel der Kasse

Schreibmaschine auf 11/2stellen Zeilenabstand einstellen!

Bestell-Nr. 80021 Kassenanordnung Umbuchungen (Flach)
Muster 65 EDVBK VwV zu § 70 SäHO
(Papier grün/Druck schwarz)
Vordruck Leitverlag GmbH, Zhl. Freiberg, Postfach 1363, 09583 Freiberg

E. Create-Anweisungen zum Erstellen der Tabellen

```
CREATE TABLE [dbo].[Artikel] (
	[ID] [int] IDENTITY (1, 1) NOT NULL ,
	[ID_Kostenart] [int] NOT NULL ,
	[Nummer] [nvarchar] (20) COLLATE Latin1_General_CI_AS NOT NULL ,
	[Bezeichnung] [nvarchar] (200) COLLATE Latin1_General_CI_AS NOT NULL ,
	[Preis] [decimal](19, 4) NOT NULL ,
	[Einheit] [nvarchar] (50) COLLATE Latin1_General_CI_AS NOT NULL ,
	[Bestand] [int] NOT NULL ,
	[Rabatt] [decimal](18, 2) NULL ,
	[Skonto] [decimal](18, 2) NULL ,
	[Zusatzinfo] [nvarchar] (250) COLLATE Latin1_General_CI_AS NULL ,
	[AngelegtDatum] [datetime] NULL ,
	[AngelegtVonID] [int] NULL ,
	[Geloescht] [bit] NULL ,
	[GeloeschtDatum] [datetime] NULL ,
	[GeloeschtVonID] [int] NULL ,
	[Unsichtbar] [bit] NULL ,
	[PreisVariabel] [bit] NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[BestellerKostenstellen] (
	[ID] [int] IDENTITY (1, 1) NOT NULL ,
	[ID_User_Stammdaten] [int] NOT NULL ,
	[Kostenstelle] [nvarchar] (8) COLLATE Latin1_General_CI_AS NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[Bestellung] (
	[ID] [int] IDENTITY (1, 1) NOT NULL ,
	[Bezeichnung] [nvarchar] (250) COLLATE Latin1_General_CI_AS NULL ,
	[ID_User_Stammdaten] [int] NOT NULL ,
	[Kostenstelle] [nvarchar] (8) COLLATE Latin1_General_CI_AS NOT NULL ,
	[DatumErstellung] [datetime] NOT NULL ,
	[Vorname] [nvarchar] (50) COLLATE Latin1_General_CI_AS NULL ,
	[Name] [nvarchar] (50) COLLATE Latin1_General_CI_AS NOT NULL ,
```

```
	[Telefon] [nvarchar] (50) COLLATE Latin1_General_CI_AS NULL ,
	[Email] [nvarchar] (50) COLLATE Latin1_General_CI_AS NULL ,
	[Anrede] [char] (6) COLLATE Latin1_General_CI_AS NULL ,
	[Titel] [char] (15) COLLATE Latin1_General_CI_AS NULL ,
	[Lieferadresse] [nvarchar] (1000) COLLATE Latin1_General_CI_AS NOT NULL ,
	[Status] [int] NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[BestellungPositionen] (
	[ID] [int] IDENTITY (1, 1) NOT NULL ,
	[ID_Bestellung] [int] NOT NULL ,
	[ID_Kostenart] [int] NOT NULL ,
	[ID_Orginal] [int] NULL ,
	[Artikelnummer] [nvarchar] (20) COLLATE Latin1_General_CI_AS NULL ,
	[Menge] [decimal](18, 0) NOT NULL ,
	[Bezeichnung] [nvarchar] (200) COLLATE Latin1_General_CI_AS NULL ,
	[Preis] [decimal](19, 4) NULL ,
	[Einheit] [nvarchar] (50) COLLATE Latin1_General_CI_AS NULL ,
	[Bemerkung] [nvarchar] (1000) COLLATE Latin1_General_CI_AS NULL ,
	[NeuerArtikel] [bit] NOT NULL ,
	[Status] [int] NOT NULL ,
	[Zeitpunkt] [datetime] NULL ,
	[Variabler_Preis] [bit] NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[Guthaben] (
	[ID] [int] IDENTITY (1, 1) NOT NULL ,
	[Kostenstelle] [nvarchar] (8) COLLATE Latin1_General_CI_AS NOT NULL ,
	[Jahr] [char] (4) COLLATE Latin1_General_CI_AS NOT NULL ,
	[Wert] [float] NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[GuthabenKoart_Kost] (
	[ID] [int] IDENTITY (1, 1) NOT NULL ,
```

```
	[Kostenstelle] [int] NULL ,
	[Kostenart] [int] NULL ,
	[Datum] [datetime] NULL ,
	[Guthaben] [float] NULL ,
	[eingetragenAm] [datetime] NULL ,
	[eingetragenVon] [int] NULL ,
	[Status] [int] NULL ,
	[Bemerkung] [nvarchar] (255) COLLATE Latin1_General_CI_AS NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[Kostenarten] (
	[ID] [int] IDENTITY (1, 1) NOT FOR REPLICATION  NOT NULL ,
	[IDparent] [int] NOT NULL ,
	[Nummer] [nchar] (20) COLLATE Latin1_General_CI_AS NOT NULL ,
	[Bezeichnung] [nchar] (100) COLLATE Latin1_General_CI_AS NOT NULL ,
	[Bemerkung] [nchar] (100) COLLATE Latin1_General_CI_AS NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[KostenstellenAusUni] (
	[inst_nr] [nchar] (10) COLLATE Latin1_General_CI_AS NOT NULL ,
	[dname] [nchar] (25) COLLATE Latin1_General_CI_AS NULL ,
	[lname1] [nchar] (50) COLLATE Latin1_General_CI_AS NULL ,
	[lname2] [nchar] (50) COLLATE Latin1_General_CI_AS NULL ,
	[buchbar] [int] NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[Lieferanten] (
	[ID] [int] IDENTITY (1, 1) NOT NULL ,
	[Name] [nvarchar] (50) COLLATE Latin1_General_CI_AS NOT NULL ,
	[Lieferantennummer] [nvarchar] (20) COLLATE Latin1_General_CI_AS NULL ,
	[Kundennummer] [nvarchar] (20) COLLATE Latin1_General_CI_AS NULL ,
	[Strasse] [nvarchar] (50) COLLATE Latin1_General_CI_AS NULL ,
	[Hausnummer] [nvarchar] (10) COLLATE Latin1_General_CI_AS NULL ,
	[PLZ] [nchar] (5) COLLATE Latin1_General_CI_AS NULL ,
```

```
	[Ort] [nvarchar] (50) COLLATE Latin1_General_CI_AS NULL ,
	[Telefon] [nvarchar] (50) COLLATE Latin1_General_CI_AS NULL ,
	[Fax] [nvarchar] (50) COLLATE Latin1_General_CI_AS NULL ,
	[Weburl] [nvarchar] (50) COLLATE Latin1_General_CI_AS NULL ,
	[Email] [nvarchar] (50) COLLATE Latin1_General_CI_AS NULL ,
	[APname] [nvarchar] (50) COLLATE Latin1_General_CI_AS NULL ,
	[APtelefon] [nvarchar] (50) COLLATE Latin1_General_CI_AS NULL ,
	[APfax] [nvarchar] (50) COLLATE Latin1_General_CI_AS NULL ,
	[APemail] [nvarchar] (50) COLLATE Latin1_General_CI_AS NULL ,
	[Rabatt] [decimal](18, 2) NULL ,
	[Skonto] [decimal](18, 2) NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[Lieferungen] (
	[ID] [int] IDENTITY (1, 1) NOT FOR REPLICATION  NOT NULL ,
	[ID_Lieferant] [int] NOT NULL ,
	[ID_User_Stammdaten] [int] NOT NULL ,
	[Lieferantenname] [nvarchar] (50) COLLATE Latin1_General_CI_AS NULL ,
	[Rechnungsnummer] [nvarchar] (50) COLLATE Latin1_General_CI_AS NULL ,
	[Rechnungsdatum] [datetime] NULL ,
	[Auftragsnummer] [nvarchar] (50) COLLATE Latin1_General_CI_AS NULL ,
	[Auftragsdatum] [datetime] NULL ,
	[Lieferscheinnummer] [nvarchar] (50) COLLATE Latin1_General_CI_AS NULL ,
	[Lieferungsdatum] [datetime] NULL ,
	[Eingabedatum] [datetime] NULL ,
	[Rabatt] [decimal](18, 2) NULL ,
	[Skonto] [decimal](18, 2) NULL ,
	[Bemerkung] [nvarchar] (200) COLLATE Latin1_General_CI_AS NULL ,
	[Gebuchtdatum] [datetime] NULL ,
	[Storniertdatum] [datetime] NULL ,
	[Status] [int] NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[LieferungenPositionen] (
	[ID] [int] IDENTITY (1, 1) NOT NULL ,
```

```
	[ID_Lieferung] [int] NOT NULL ,
	[ID_Artikel] [int] NOT NULL ,
	[ArtNumLieferant] [nvarchar] (50) COLLATE Latin1_General_CI_AS NULL ,
	[Bezeichnung] [nvarchar] (200) COLLATE Latin1_General_CI_AS NULL ,
	[Menge] [int] NOT NULL ,
	[Zugang] [nvarchar] (50) COLLATE Latin1_General_CI_AS NULL ,
	[Preis] [decimal](19, 2) NOT NULL ,
	[Einheit] [nvarchar] (50) COLLATE Latin1_General_CI_AS NULL ,
	[Rabatt] [decimal](18, 2) NULL ,
	[Skonto] [decimal](18, 2) NULL ,
	[Zusatzinfo] [nvarchar] (250) COLLATE Latin1_General_CI_AS NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[User_Login] (
	[ID] [int] IDENTITY (1, 1) NOT NULL ,
	[Login] [nvarchar] (50) COLLATE Latin1_General_CI_AS NOT NULL ,
	[Passwort] [nvarchar] (50) COLLATE Latin1_General_CI_AS NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[User_Stammdaten] (
	[ID] [int] NOT NULL ,
	[ID_User_Type] [int] NOT NULL ,
	[ID_inst_nr] [nvarchar] (8) COLLATE Latin1_General_CI_AS NULL ,
	[Vorname] [nvarchar] (50) COLLATE Latin1_General_CI_AS NULL ,
	[Name] [nvarchar] (50) COLLATE Latin1_General_CI_AS NOT NULL ,
	[Email] [nvarchar] (50) COLLATE Latin1_General_CI_AS NOT NULL ,
	[Telefon] [nvarchar] (50) COLLATE Latin1_General_CI_AS NULL ,
	[Anrede] [char] (6) COLLATE Latin1_General_CI_AS NOT NULL ,
	[Titel] [char] (15) COLLATE Latin1_General_CI_AS NULL ,
	[Geloescht] [bit] NOT NULL ,
	[showFakultaet] [int] NULL ,
	[Extern] [int] NULL
) ON [PRIMARY]
GO
```

```
CREATE TABLE [dbo].[User_Type] (
	[ID] [int] IDENTITY (1, 1) NOT NULL ,
	[Bezeichnung] [nvarchar] (50) COLLATE Latin1_General_CI_AS NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[Warenkorb] (
	[ID] [int] IDENTITY (1, 1) NOT FOR REPLICATION  NOT NULL ,
	[ID_User_Stammdaten] [int] NOT NULL ,
	[Kostenstelle] [nvarchar] (8) COLLATE Latin1_General_CI_AS NOT NULL ,
	[Wk_Titel] [nvarchar] (100) COLLATE Latin1_General_CI_AS NOT NULL ,
	[Status] [int] NOT NULL ,
	[Lieferadresse] [nvarchar] (1000) COLLATE Latin1_General_CI_AS NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[WarenkorbPositionen] (
	[ID] [int] IDENTITY (1, 1) NOT FOR REPLICATION  NOT NULL ,
	[ID_Warenkorb] [int] NOT NULL ,
	[ID_Kostenart] [int] NOT NULL ,
	[Artikelnummer] [nvarchar] (20) COLLATE Latin1_General_CI_AS NULL ,
	[Menge] [decimal](18, 0) NOT NULL ,
	[Bezeichnung] [nvarchar] (200) COLLATE Latin1_General_CI_AS NULL ,
	[Preis] [decimal](18, 4) NULL ,
	[Einheit] [nvarchar] (50) COLLATE Latin1_General_CI_AS NULL ,
	[Bemerkung] [nvarchar] (1000) COLLATE Latin1_General_CI_AS NULL ,
	[NeuerArtikel] [bit] NOT NULL
) ON [PRIMARY]
GO
```

Erklärung

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

Leipzig, Januar 2003 .....

Unterschrift