Bauhaus-Universität Weimar

Fachbereich: Informatik im Bauwesen

Schwerpunkt: plattformunabhängige Schnittstellen

Studienarbeit zum Thema:

Entwurf einer plattformunabhängigen Schnittstelle für den Zugriff auf die Standardleistungsbuchtexte in „STLB-Bau“

Sommersemester 2000

vorgelegt bei:

Prof. Dr. Ing. K. Beuke

Dr.-Ing. H. Kirschke

von:

Marco Heinßmann

Ferdinand-Freiligrath-Str. 8

99423 Weimar

Tel.: 03643/901537

E-Mail: marco.heinszmann@bauing.uni-weimar.de

marco.heinszmann@web.de

INHALTSVERZEICHNIS

ABKÜRZUNGSVERZEICHNIS

COM („Component Object Model“)

Komponententechnologie von Microsoft; wiederverwendbare Softwarekomponenten auf binärer Ebene.

DLL („Dynamic Link Library“)

DLL’s stellen eine einfache Art von Komponentenbildung dar. In DLL’s können Klassen eingebunden werden, auf die andere Programme zurückgreifen können. Verfügbar ist dieser Mechanismus allerdings nur auf MS-Windows-Plattformen.

http („HyperText Transfer Protocol“)

Client-Server-TCP/IP-Protokoll, das im WWW oder Intranets für den Austausch von HTML-Dokumenten benutzt wird. Die Standardisierung von HTTP obliegt dem W3C.

HTTPS („HTTP Secure“)

sicheres HTTP; Ist eine Variante von HTTP für sichere Transaktionen. Browser, die HTTPS unterstützen, verwenden "https://", um Zugriffe auf URLs auf HTTP-Servern mit Hilfe von SSL zu ermöglichen. HTTPS gilt als eigenes Protokoll besteht aber faktisch aus einem HTTP mit einem SSL darunter.

SSL (“Secure Sockets Layer”)

Ein von der Firma Netscape Communications geschaffenes Protokoll zu Schaffung einer sicheren (verschlüsselten) Kommunikation im Internet. SSL bildet eine Schicht zwischen den Applikationsprotokollen wie ua HTTP und dem Transportprotokoll TCP/IP. Hauptsächlich wird es von HTTPS benutzt.

Die durch SSL verwendete Verschlüsselung bietet Schutz gegen die meisten Datenangriffe.

XML ("eXtensible Markup Language"; erweiterbare Auszeichnungssprache).

Ende 1997 vom WWW Consortium vorgestellte Beschreibungssprache, die eine reduzierte Variante von SGML darstellt. XML wurde als professionelle Alternative zu HTML konzipiert und zeichnet sich dadurch aus, dass sich eigene Befehle definieren lassen, welche Sprache um Fähigkeiten erweitern, die HTML nicht bieten kann.

1. EINFÜHRUNG

1.1 EINLEITUNG

Die anhaltende Krise in der Bauwirtschaft hat einen für viele Bauunternehmen ruinösen Preiswettbewerb zur Folge. Es fehlt an Aufträgen, Impulsen und Investitionen. Produktivitätssteigerungen sind daher zwingend nötig und gerade in der Bauwirtschaft gibt es genügend Ansatzpunkte dafür. Diese Ansatzpunkte liegen meist weniger in neuen Herstellungstechnologien, sondern in einer verbesserten Logistik des gesamten Herstellungsprozesses, da dieser sehr datenintensiv und arbeitsteilig ist. Immer mehr geht es auch darum, den gesamten Lebenszyklus eines Bauobjektes von der Planung bis zum Betrieb oder evtl. sogar bis zum Abriss zu optimieren.

Die Datenintensivität und Arbeitsteilung sind also die Hauptansatzpunkte einer Optimierung. Datenintensivität allein ist im Zeitalter der Informationstechnologien und immer weiter wachsender Rechnerleistung kein gravierendes Problem. Problematischer ist der Datenaustausch zwischen den am Bau Beteiligten, der infolge der hohen Arbeitsteilung notwendig ist.

Durch den Mangel an etablierten Standardschnittstellen ist es immer noch üblich, Kommunikation in Papierform zu betreiben. Dies verursacht lange Lauf und Reaktionszeiten. Redundanz in den Datenbeständen und Informationsverluste bestimmen das Bild.

Ein wichtiger Schritt in Richtung einer Optimierung ist das neue Standardleitungsbuch im Bauwesen. Es kann im Falle einer Etablierung als Standard als Systemkette fungieren - als ein Bindeglied zwischen Entwurf, Planung, Ausschreibung und Bauausführung.

1.2 PROBLEMSTELLUNG

Die fortschreitende weltweite Vernetzung und die Erschließung des Internets für neue Anwendungsbereiche, die u.a. auch eine Optimierung von Geschäftsprozessen ermöglicht schreitet immer weiter fort. Auch das Baugewerbe, das sich traditionell durch etablierte Strukturen und etwas längere Entscheidungswege auszeichnet, sollte sich dieser Entwicklung nicht entziehen.

Besondere Ansatzpunkte liegen hier in der Optimierung und Standardisierung des Datenaustausches zwischen den am Bauprozess Beteiligten, wie es bereits in der Einleitung angesprochen wurde.

Das schon genannte STLB-Bau liegt bisher nur in einer Implementationsform vor, die keine Nutzung über das Internet ermöglicht. Ein erster naheliegender Schritt in diese Richtung ist also die Verfügbarmachung der STLB-Bau-Daten

über das Internet und die Erschließung von Möglichkeiten diese Daten auch über das Internet weiter zu nutzen. In diese Richtung zielt deshalb auch die Problemstellung dieser Arbeit. Besonders die zur Verfügung stehenden Techniken der informationstechnischen Realisierung sollen dabei untersucht werden. Dies ist nötig, um später eine Implementierung und damit konkrete Umsetzung vorzubereiten.

Da die Internet-Bereitstellung der STLB-Bau-Daten bereits von deren Entwicklern geplant ist, soll der Schwerpunkt hinsichtlich der Datennutzung über das Internet bestehen. Dazu ist es allerdings trotzdem nötig, die Struktur der Daten und den Mechnanismus der Internet-Verfügbarmachung zu kennen.

1.3 Gliederung der Arbeit

Nach einer Einführung in Kapitel 1 soll in Kapitel 2 zunächst ein kurzer Einblick in die Geschichte des Standardleistungsbuches, das Standardleistungstexte für das Baugewerbe enthält, gegeben werden. Es wird klar, dass die überwiegenden Nachteile der alten Leistungstextsammlung, die in Kapitel 2.1 beschrieben werden, einer Überarbeitung und Neufassung bedingten. Das völlig neue Konzept und die Eigenschaften des neuen Standardleistungsbuches sollen dann in Abschnitt 2.2 kurz herausgestellt werden.

In Kapitel 3 erfolgt dann eine Analyse hinsichtlich der informationstechnischen Realisierung des STLB-Bau. Vor- und Nachteile werden kurz erläutert. Im Zeitalter der zunehmenden Nutzung des Internets auch für Geschäftsprozesse, wird deutlich, dass es sinnvoll ist, auch STLB-Bau-Daten im Internet verfügbar zu machen.

In den Kapiteln 4.1 und 4.2 wird kurz auf allgemeine Prinzipien und Anforderungen solcher Online-Applikationen eingegangen. Die Client-Server-Architektur spielt hier eine wichtige Rolle.

In Kapitel 4.3 soll kurz auf den Server eingegangen werden, der von den Entwicklern der STLB-Bau-Daten geplant ist. Dieser Server soll durch eine offene Schnittstelle, auch anderen Entwicklern die Möglichkeit bieten, Zugriff auf die Daten zu haben und eigene Anwendungen zu entwickeln.

Dazu ist zunächst ein STLB-Bau-Client- erforderlich, auf den in Kapitel 4.4 eingegangen wird. Bevor für diesen Client erste Realisierungsvorschläge in Kap. 5 gemacht werden, sollen noch die möglichen Strukturen von Anwendungen zur Weiterverarbeitung der Daten dieses Clients angesprochen werden.

Auf online-fähige Realisierungsformen von AVA-Systemen, also die Weiternutzung der STLB-Bau-Daten, soll zum Schluss eingegangen werden. Erste Realisierungsschritte und die zur Verfügung stehenden Technologien sollen dabei im Vordergrund stehen.

2. GESCHICHTE UND INHALT DES STLB-BAU

2.1 DAS ALTE STLB-BAU – STATISCHE BAUDATEN

Seit 1967 existiert das vom *Gemeinsamen Ausschuss für Elektronik im Bauwesen GAEB* herausgegebene Standardleistungsbuch. Dessen Aufgabe ist es, Beschreibungen von Bauleistungen klar und eindeutig zu formulieren. Für die öffentliche Hand, die nach §9 Abs. 5 der VOB produkt- und herstellerneutral ausschreiben muss, erfüllte dieser Standard lange Zeit seine Aufgaben.

Das alte *STLB-Bau* kann als eine Sammlung von vorformulierten Leistungstexten angesehen werden, also *statischen Baudaten*. Es lässt sich leicht abschätzen, dass sich bei der Vielzahl von Bauleistungen und deren Varianten ein immer größer werdender Datenbestand entwickelte. Geschätzt wird eine Anzahl von ca. 100 Mio. Leistungsvarianten. Ein kaum aktualisierbarer Bestand von Stammdaten, der einen enormen Ressourcenbedarf hatte und in vielen Punkten an seine Grenzen stieß. Ein Punkt in der Benutzung war z.B. die Suche nach dem Text einer bestimmten Leistungsbeschreibung. Sollte eine exakter Text vorliegen, so war es nicht selbstverständlich diesen auch zu finden. Der Umgang mit diesen Texten bedarf deshalb einiger Erfahrung und Zeit. Ein anderes Problem ist die Aktualisierbarkeit solcher statischen Texte. Auch wenn sich Bauleistungen in ihrer Art nicht so stark verändern, wie Leistungen in anderen Wirtschaftsbereichen, sammelten sich eine Reihe von Lücken im Datenbestand an.

Von der freien Wirtschaft dagegen wurde dieser Standard allerdings weitestgehend nicht angenommen, da sie die Formulierungsfreiheit eingeschränkt sah. Häufig sah es in der Praxis so aus, dass ähnliche Leistungstexte nur angepasst wurden, um der exakten Beschreibung einer Bauleistung möglichst nahe zu kommen. So bestimmen heute noch Unschärfen in den Texten die Ausschreibungspraxis und nicht selten wird der Optik einer Ausschreibung mehr beigemessen als deren Inhalt. Deshalb startete der *Gemeinsame Ausschuss für Elektronik im Bauwesen* 1994 die Ausschreibung eines *STLB*-Nachfolgers.

Dieser sollte u.a. Leistungsbeschreibungen *VOB*- und *DIN*-gerecht abbilden, alle Leistungsbereiche abdecken und die Einbindung neuer fachlicher Zielsetzungen erlauben. Um das Datenaufkommen zu reduzieren und überschaubar zu halten, welche Bedingungen für eine Aktualisierbarkeit sind, wurde eine Leistungsbeschreibung auf Basis von Teilleistungen gefordert. Eine völlig neue Technologie soll diese Teilleistungen dann zu kompletten Leistungstexten zusammensetzen. Den Daten also einen dynamischen Charakter verleihen.

2.2 DAS NEUE STLB-BAU – DYNAMISCHE BAUDATEN

Im März 1996 fiel die Entscheidung der Ausschreibung zu Gunsten der *Dr. Schiller & Partner GmbH,* die neben 10 anderen Softwarehäusern ein Angebot machte. Seit Oktober 1996 wird das neue Standardleistungsbuch unter dem Namen *STLB-Bau Dynamische Baudaten* von DIN angeboten und systematisch erweitert. Anfang 2000 soll das alte STLB bei der öffentlichen Hand völlig abgelöst werden. Die schon im Namen enthaltene Dynamik der Baudaten und deren andere vielfältige Möglichkeiten sollen im folgenden beschrieben werden.

Schon bei der Ausschreibung wurde gefordert, Leistungsbeschreibungen auf Basis von Teilleistungen abzubilden. Diese Forderung wurde realisiert, indem die vorhanden Volltexte in Textteile aufgelöst und systematisch strukturiert wurden. So konnten charakteristische Beschreibungsmerkmale gefunden werden. Zusätzlich wurden Varianten generiert. Diese sind Beschreibungsmerkmale einer Teilleistung mit deren Ausprägungen. Diese Ausprägungen stellen eine exakte Spezifikation des Beschreibungsmerkmals dar. Abb.1 zeigt beispielhaft einen kompletten Leistungstext. In Abb. 2 ist die Aufschlüsselung dieses Leistungstextes in Beschreibungsmerkmale und Ausprägungen zu sehen. Ebenfalls entspricht die Reihenfolge der Abbildung in etwa der Abfolge des interaktiven Dialogs, mit dem diese Daten zusammengestellt werden.

m2 Mauerwerk
Innenwand
Einseitig als Sichtmauerwerk, Verfugen wird gesondert vergütet. Mauerziegel DIN 105-Mz 20-2.0 NF (240*115*71), MGII,
Wandstärke 11,5cm

Abb. 1
Beispiel einer Leistungsbeschreibung

Beschreibungsmerkmale		**Ausprägungen**
Mauerarbeiten		
Mauerwerk Mauersteine		
Wände Mauerziegel	.....	Innenwand
Bauteil, Wand/aufgehend		
Dicke: ***	.....	11,5 cm
Mauerstein:	.....	DIN 105-Mz 20-2.0
Format:	.....	NF (240x115x71)
Mauermörtel:	.....	MGII
Anforderung Mauerwerk:	.....	Einseitig als Sichtmauerwerk
Fugenbearbeitung Mauerwerk:		Verfugen wird gesondert vergütet
Abrechnungseinheit:	.....	m2

Abb. 2
Aufgliederung einer Leistungsbeschreibung in Beschreibungsmerkmale und Ausprägungen

Durch die intelligente Zuordnung verschiedener Varianten zu ihren Teilleistungen entsteht dann die komplette eindeutige Leistungsbeschreibung. Diese entspricht durch den vorgegeben Zuordnungsmechanismus den gültigen Normen und Bestimmungen. So kann z.B. in einem einfachen Fall ausgeschlossen werden, dass die Innenwand in Abb. mit 11,5 cm Wanddicke aus Ziegeln des Formats 24x36.5x23.8 ausgeschrieben wird.

Die Struktur von Varianten und deren Ausprägungen hat allerdings noch eine Vielzahl anderer Vorteile. Durch die Codierung jeder Leistungsvariante wird es möglich, nicht nur die bloße Textinformation zu erhalten. Ein Beispiel einer solchen Codierung ist in Abb. 3 enthalten.

Identifikator des Beschreibungs-merkmals	Bezeichnung des Beschreibungs-merkmal	Identifikator der Ausprägung	Bezeichnung der Ausprägung	Schlüssel der Ausprägung	
39	Wanddicke	5	11,5 cm	39	5
39	Wanddicke	6	15 cm	39	6
39	Wanddicke	7	17,5 cm	39	7
39	Wanddicke	8	24 cm	39	8

Abb. 3

Ausprägung und deren Codierung als konkrete Angabe zu einem Beschreibungs-Merkmal

nach [1]

Jedes Beschreibungsmerkmal sowie jede Ausprägung besitzt einen Indentifikator. Diese Codierung erlaubt auch eine eindeutige Identifizierung des Leistungstextes und das Anhängen von beliebigen Informationen an diese Codierung und damit auch an die komplette Teilleistung. So kann z.B. durch Kalkulationsansätze an jeder Ausprägung die konstruktive Entstehung von Teilleistungen aus ihren Einzelkosten nachvollzogen werden. Dies ermöglicht z.B. das verursachungsgerechte Verfolgen der Kosten. Andere Informationen wie DIN- und VOB-Regeln, bauphysikalisches Wissen, Produktinformationen u.a. sind nur einige Beispiele der realisierbaren Möglichkeiten. Die STLB-Bau-Daten stellen allerdings immer die zentrale Informationsquelle dar. Ein Beispiel für den Code einer kompletten Teilleistung kann Abb.4 entnommen werden.

Im Falle einer Etablierung des neuen STLB-Bau in der Bauwirtschaft besteht die Möglichkeit, dem Ziel einer durchgängigen abgesicherten Informationskette für alle am Bau Beteiligten, einen wichtigen Schritt näher zu kommen. Die neue GAEB2000-Schnittstelle sorgt dabei, wie die früheren Versionen, für einen standardisierten Datenaustausch.

Durch die Aufspaltung in Teiltexte ergeben sich eine Reihe von Vorteilen. So benötigen die Daten nur geringe Ressourcen, redundante Datenbestände können vermieden und daduch die Aktualisierbarkeit wesentlich erleichtert werden.

```
2110100010 NNN         00000001000m2                                          0(
2210100010  0000000000  000000000000                                           0(
79STLB-Bau  KAT STLB-Bau VJ 1999 VM 10 LB 012 TLG 58 BM 2795 BMI 0 AP 20 000012
79STLB-Bau  BM 39 BMI 0 AP 11 BM 2 BMI 0 AP 1 BM 6 BMI 0 AP 3 BM 3 BMI 0 000013
79STLB-Bau  AP 5 BM 4 BMI 0 AP 23 BM 5 BMI 0 AP 21 BM 543 BMI 0 AP 5 BM  00001
79STLB-Bau  3056 BMI 0 AP 10 BM 2287 BMI 0 AP 21 BM 2501 BMI 0 AP 22 BM  00001
79STLB-Bau  2142 BMI 0 AP 77 BM 7691 BMI 0 AP 1 BM 10065 BMI 0 AP 2 BM   0000
79STLB-Bau  2136 BMI 0 AP 1                                                 0(
25                                                                          0(
26   Mauerwerk DIN 1053-1 der Innenwand, einseitig als              000019
26   Sichtmauerwerk, Mauerziegel, DIN 105, Mz,                      000020
26   Festigkeitsklasse 20, Mauerwerksdicke 11,5 cm, MG II,          000021
26   NF (240 x 115 x 71).   ...............                         000022
```

Abb. 4

Teilleistung , wie sie in GAEB – Schnittstelle abgebildet wird.

Wie diese Lösung informationstechnisch realisiert wurde, soll im folgenden beschrieben werden.

3. ANALYSE DES BESTEHENDEN SYSTEMS

3.1 ARCHITEKTUR DES BESTEHENDEN SYSTEMS

STLB-Bau besteht aus dem Programmmodul oder auch Auswahldialog und den Daten. Das Programmmodul, dass als DLL realisiert ist, kann von beliebigen Windows-Applikationen aufgerufen werden. Dieses Programmmodul enthält die nötige Funktionalität des Zugriffs auf die Datenbank und die Logik der Zusammenstellung der Leistungstexte sowie die Benutzerführung des Auswahldialogs. Zusätzlich müssen Daten von der Programmschnittstelle übernommen bzw. an diese übergeben werden. In Abb. 14 ist u.a. dieses Programmodul gezeigt.

Die STLB-Bau-Daten enthalten Beschreibungsmerkmale, Ausprägungen, Teilleistungsgruppen, Schlagworte, Leistungsbereiche und deren Beziehungen untereinander.

Da dieses Programmmodul als DLL vorliegt kann sich jeder Entwickler dieses Moduls bedienen und es in seine Applikation einbinden. So wird sichergestellt, dass AVA-Programmen der Zugriff auf STLB-Bau-Daten möglich ist. Diese aufrufenden Programme steuern dieses Programmmodul über eine Schnittstelle. Über diese Schnittstelle erfolgt nicht nur der gesammte Datenaustausch zum und vom Programmmodul sondern auch dessen Initialisierung und Beendigung. Abb.5 zeigt diesen Sachverhalt.

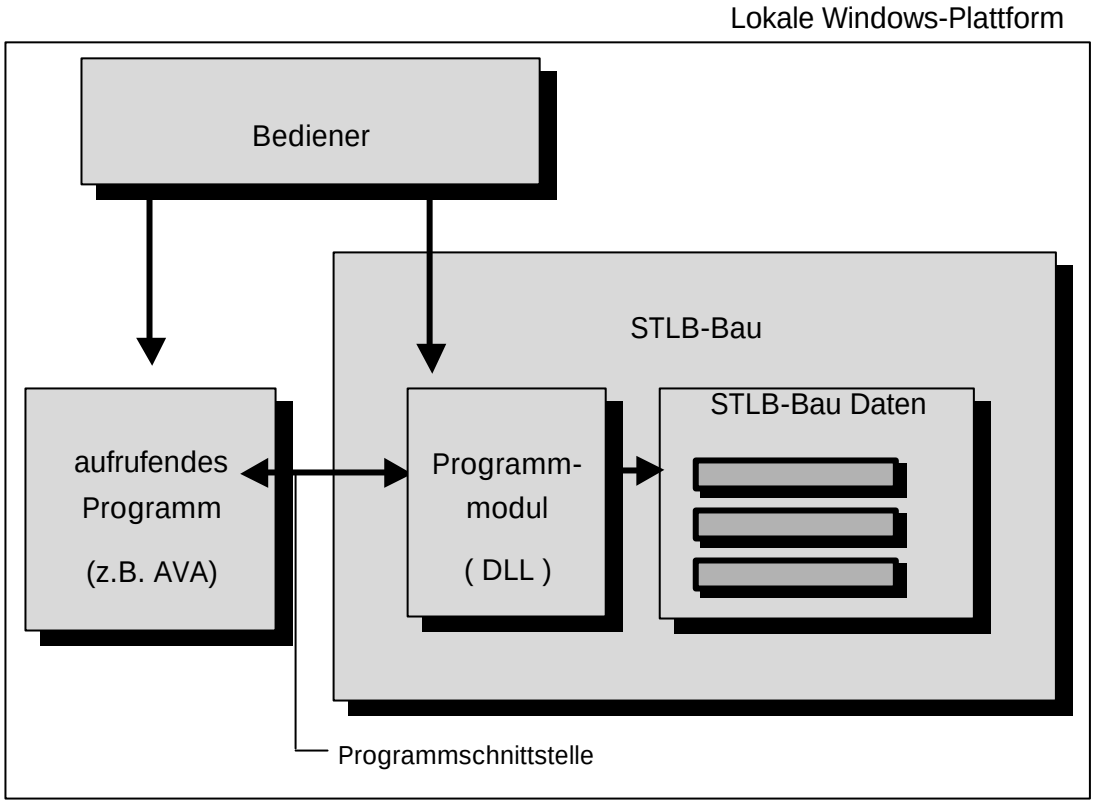

Abb. 5

Zusammenwirken von Bediener, aufrufendem Programm und STLB-Bau

nach [1]

3.2 DATEN AN DER PROGRAMMSCHNITTSTELLE

An der Programmschnittstelle werden alle Daten bereitgestellt, die für die weitere Bearbeitung einer Teilleistung außerhalb von STLB-Bau benötigt werden. Die Tabelle in Abb.6 listet diese Daten auf, in Abb.7 erfolgt eine kurze Beschreibung.

Datenelemente	Beispiel		
Mengeneinheit	m3		
Positionskurztext	Außenwand . . .		
Liste: Zeilen Positionslangtext	Mauerwerk der Außenwand . . .		
	nächste Zeile Positionslangtext		
	letzte Zeile Positionslangtext		
Leistungsbereichskürzel	012		
Leistungsbereichsbezeichnung	Mauerarbeiten		
Identifikator TLG	17		
Liste: Schlüssel für Ausprägungen	39	0	17
	246	0	3
	4	0	82
	56	0	1
	34	0	11
Katalogbezeichnung	STLBBau		
Jahr der Version	1998		
Monat der Version	10		

Abb. 6

Daten an der Programm-schnittstelle

nach [1]

- eine Mengeneinheit
- eine Zeile Positionskurztext (Länge kann zur Laufzeit abgefragt werden)
- eine Liste mit Zeilen Positionslangtext (max. 55 + 1 Zeichen je Zeile)
- den Langtext im RT-Format (Länge kann zur Laufzeit abgefragt werden)
- einen Identifikator der Teilleistungsgruppe
- eine Liste mit den Schlüsseln der Ausprägungen
- ein Leistungsbereichskürzel (max. 3+1 Zeichen)
- eine Leistungsbereichsbezeichnung (max. 55 + 1 Zeichen)
- Katalogbezeichnung (max. 10 + 1 Zeichen)
- Jahr der Version des Kataloges (genau 4 + 1 Zeichen)
- Monat der Version des Kataloges (genau 2 + 1 Zeichen)
- Kennung des STLB-Bau Textes (max. 55 + 1 Zeichen)

Abb. 7

Daten an der Programm-schnittstelle

nach [1]

Die Auflistung dieser Daten soll schon an dieser Stelle erfolgen, da absehbar ist, dass diese hinsichtlich der Entwicklung einer plattformunabhängigen Schnittstelle benötigt werden.

3.3 Vorteile und Nachteile des bestehenden Systems

Mit dem bestehenden STLB-Bau-System steht ein gut funktionierendes System bereit, das sehr schnellen Zugriff auf die Wissensdatenbank erlaubt. Über den einheitlichen STLB-Bau-Auswahldialog und die zur Verfügung stehende Programm-Schnittstelle ist es vielen Entwicklern von AVA-Programmen relativ einfach möglich, auf STLB-Bau-Daten zuzugreifen. Das bestehende System besitzt allerdings die für lokale, plattformabhängige Installationen typischen Nachteile.

Die STLB-Bau-Daten stehen nur denjenigen zur Verfügung, die mit einer [MS]-Windows–Plattform arbeiten. Da AVA-Programme, die unter anderen Systemen arbeiten einen Marktanteil von nur ca. 2% besitzen, mag dieser Sachverhalt zunächst nicht so wichtig erscheinen. Warum soll sich aber der Status, den Windows besitzt, in der Zukunft nicht ändern?

Ein zunächst schwerwiegender erscheinender Nachteil zeigt sich in der Aktualisierbarkeit einer lokalen Installation. Derzeit muss jede Einzelplatzinstallation regelmäßig mit Updates in Form von Downloads oder durch Beziehen einer CD-ROM versorgt werden. Dies stellt einen nicht unerheblichen zeitlichen und wirtschaftlichen Aufwand dar.

Der wohl größte Nachteil ist, dass jeder Nutzer, der die Daten verwenden möchte, eine Lizenz erwerben muss. Dieser finanzielle Aufwand ist nur sinnvoll, wenn die Daten in entsprechenden Mengen genutzt werden. Ebenfalls kauft der Nutzer Daten, von denen er häufig nur einen Bruchteil benötigt. Diese Tatsache verschließt die STLB-Bau-Daten einer großen potentiellen Nutzergruppe.

Deutlicher werden diese Punkte noch, wenn man STLB-Bau-Daten mit den in Kapitel 2.2 beschriebenen Zusatzinformationen wie z.B. Produktdaten ausstatten möchte. Wenn auf einer lokalen Installation zusätzlich noch Herstellerkataloge oder andere ständig zu pflegenden Daten verwendet werden sollen, steigt der zeitliche und wirtschaftliche Aufwand enorm an.

Alle diese Probleme könnten sich durch eine funktionierende onlinefähige Zugriffslösung auf die Wissensbank des STLB-Bau lösen lassen. Die Zugriffssysteme wären bei plattformunabhängiger Implementierung nicht auf MS-Windows-Systeme beschränkt. Es müsste nur eine zentrale Datenbank aktualisiert und gepflegt werden. Außerdem besteht die Möglichkeit durch vielfältige individuelle Anmeldungs- und Abrechnungssysteme völlig neue Dienstleistungen für die Bauwirtschaft bereitzustellen. Aus diesen Gründen soll es das Ziel sein, in den folgenden Kapiteln Möglichkeiten für einen solchen Online-Zugriff zu suchen.

4. Ziel einer Online-Nutzung der Daten

4.1 Client-Server-Prinzip

Als Grundprinzip für netzwerkfähige Anwendungen kann die Client-Server-Architektur angesehen werden. Clients laufen auf sog. Clientmaschinen und stellen Anfragen an den Server. Diese Requests stellt der Client, da er aus unterschiedlichsten Gründen nicht in der Lage ist, die anstehenden Aufgaben eigenständig zu bearbeiten.

Ein Server ist ein Subsystem auf einer Maschine, welches nach außen hin obige Dienste anbietet und die *Requests* der Clients bearbeitet. Server sind dabei kontinuierlich ablaufende Prozesse, die auf mögliche Anforderungen (Requests) der Clients warten.

In Abb. 8 wird dieses Client-Server-Prinzip auf die zu bearbeitende Problemstellung eines STLB-Bau-Clients bzw. –Servers bezogen. Dies geschieht, indem die bisherige Systemstruktur, die in Abb.5 gezeigt wird, um einige Elemente erweitert wird. Als wichtigste Erweiterung ist das Datenzugriffsmodul anzusehen. Bei der bisherigen Lösung übernimmt das Programmmodul alle Aufgaben, sowohl den Datenbankzugriff, Anzeige der Daten, die Logik der Bedienerführung als auch die Schnittstellenfunktionen zum aufrufenden Programm.

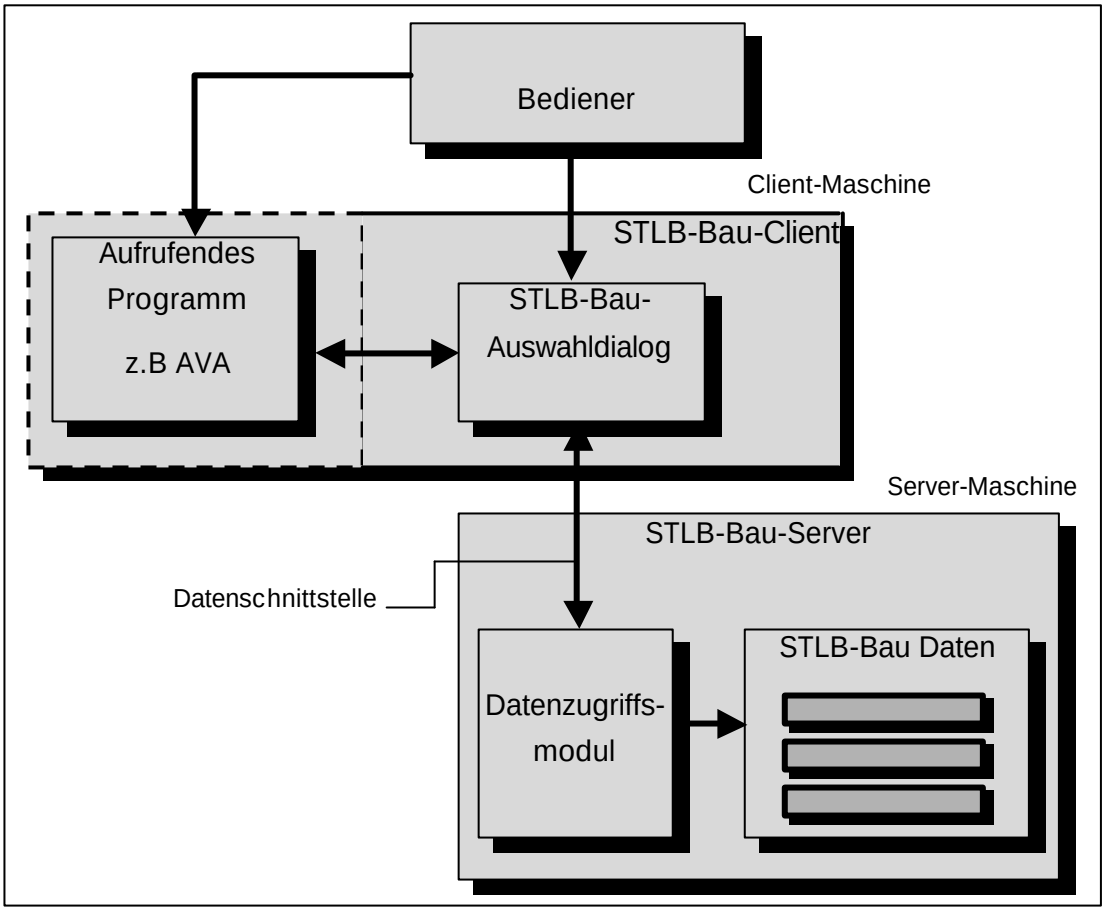

Abb. 8

Erweiterung von Abb. 5 hinsichtlich einer Client-Server-Architektur

Um aber das Client-Server-Prinzip zu verwirklichen, bedarf es der Aufteilung dieser Aufgaben. Dazu wird das Datenzugriffsmodul verwendet. Es hat die Aufgaben, auf die Daten zuzugreifen, die Steuerung der Bedienerführung zu übernehmen. Der STLB-Bau-Auswahldialog übernimmt die anderen Aufgaben, nämlich Anzeige der Daten, Bedienungsfunktionen und die Schnittstellenfunktionen zum aufrufenden Programm. Nätürlich müssen diese beiden Komponenten auch über eine Schnittstelle kommunizieren, die in Abb.8 als Datenschnittstelle bezeichnet wird. Zusammengefasst kann also das Datenzugriffsmodul mit der angeschlossenen Datenbank als Server und der STLB-Bau-Auswahldialog als Client bezeichnet werden. Beide laufen auf den bereits genannten Server- bzw. Client-Maschinen.

Das aufrufende Programm, besitzt vorerst eine untergeordnete Rolle, da es hierfür wieder eine Reihe von verschiedenen Realisierungsfomen gibt. Allen gemeinsam ist, dass der STLB-Bau-Auswahldialog wiederum als ein Server und das aufrufende Programm als Client betrachtet werden können, da der Auswahldialog das aufrufende Programm mit Daten versorgt. Aus diesem Grund ist das aufrufende Programm auch von der Client-Maschine abgekoppelt.

4.2 Allgemeine Anforderungen an Online-Applikationen

Internet-basierte Lösungen stellen gegenüber lokalen Installationen wesentlich größere Ansprüche hinsichtlich der Skalierbarkeit, Offenheit, Zuverlässigkeit, Verwaltbarkeit, Performance und vor allem Sicherheit als konventionelle Anwendungen. Im folgenden soll kurz auf die wichtigsten allgemeinen Anforderungen eingegangen werden, bevor in Kapitel 5.1 die speziellen Anforderungen des zu entwickelnden Systems erläutert werden.

4.2.1 Skalierbarkeit

Skalierbare Systeme haben die Eigenschaft, ihre Größe ohne Veränderung von System- und Anwendersoftware modifizieren zu können. So sind z.B. die Anzahl der Benutzer, Workstations, Servermaschinen, Clients in ihrer Anzahl variabel und den aktuellen Erfordernissen anpassbar. Spezielle Problembereiche sind dabei zentrale Tabellen, Komponenten und Algorithmen, da diese insbesondere bei zunehmenden Benutzerzahlen auch zunehmend beansprucht werden. Dies sind natürlich Anforderungen, die den serverseitigen Teil der Applikation betreffen.

4.2.2 Zuverlässigkeit

Der Aspekt der Zuverlässigkeit betrifft hauptsächlich den serverseitigen Teil des Systems. Dessen Verfügbarkeit kann z.B. durch eine Verteilung erhöht werden.

Bei einem Ausfall eines Teils des Systems kann das Restsystem theoretisch, mit verminderter Leistung weiterarbeiten.

Natürlich ist auch für den clientseitigen Teil der Online-Applikation eine große Zuverlässigkeit zu garantieren. Ein maßgeblicher Gesichtspunkt ist hier die Verwendung allgemeingültiger und nicht herstellungsspezifischer Implementierungsstandards oder -richtlinien. Diese sollen auch die Offenheit des Systems gegenüber funktionalen Erweiterungen und dem Einsatz auf verschieden Plattformen erhöhen.

4.2.3 Sicherheit

Bei Online-Applikationen ist der Sicherheit besondere Beachtung zu schenken, da diese in einer offenen, nahezu unkontrollierbaren Umgebung ablaufen. Besonders wichtig ist der Schutz gegen unbefugten Zugriff auf eigene Daten oder deren Manipulation sowie der Schutz vor unbefugter Ressourcenreservierung. Ein Aspekt der Sicherheit sind ebenso Verpflichtungen, die bei der Benutzung einer kommerziellen Anwendung notwendigerweise eingegangen werden.

4.3 Server für STLB-Bau-Daten

Die Entwickler der STLB-Bau-Daten verfolgen das Ziel, einen offenen Zugriff auf die Wissensbank zu gewährleisten. Der Server, der diesen Dienst anbietet, muss also die gesamte Funktionalität besitzen, die nötig ist, um Leistungstexte des STLB-Bau daten- und logikmäßig für den STLB-Bau-Auswahldialog als Client zusammenzustellen. Diese Aufgabe übernimmt das Datenzugriffsmodul mit der angeschlossenen Datenbank, das in Kapitel 4.1 schon kurz beschrieben wurde.

Abb. 9 zeigt das Modell der serverseitigen Bereitstellung der Daten, wie es von den Entwicklern des STLB-Bau geplant ist. Da sich die bestehende Datenbanklösung hinsichtlich den Anforderungen von Skalierbarkeit und Zuverlässigkeit bewährt hat, soll diese bestehen bleiben. Das Datenzugriffsmodul arbeitet auf dieser STLB-Bau-Datenbank. Dieses Datenzugriffsmodul soll in Komponentenform realisiert werden. Der COM-Standard für Komponentensoftware soll hierbei zum Einsatz kommen. Dies bedingt natürlich MS-Windows als Serverbetriebssystem.

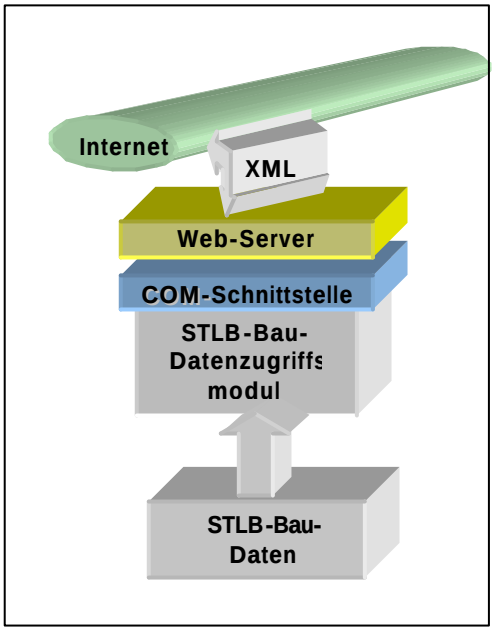

Abb. 9

Modell des STLB-Bau-Servers und der Daten-Bereitstellung

Der Web-Server kann also durch die definierte Schnittstelle mit diesem Datenzugriffsmodul kommunizieren. Das Datenzugriffsmodul ist also für STLB-Bau-spezifische Aufgaben notwendig und der Web-Server regelt die Internetanbindung und den allgemeinen Datenverkehr.

Wichtiger für diejenigen, die über das Internet auf diese Daten zugreifen möchten, ist allerdings die geplante Schnittstelle, da diese offengelegt werden soll und Entwicklern die Möglichkeit bieten soll, eigene Lösungen auf der Basis der STLB-Bau-Daten zu entwickeln. Einige Ansätze für solche Anwendungsmöglichkeiten wurden in Kapitel 2.2 genannt.

Der jetzige Stand der Entwicklung dieser Schnittstelle lässt noch keine zuverlässigen Aussagen zu deren genauen Funktionsweise oder zu. Fest steht nur, dass diese auf Basis von XML definiert sein wird. Auf diese Technologie des Datenaustausches soll später noch weiter eingegangen werden.

4.4 Clients für STLB-Bau-Daten

Bevor Clients für STLB-Bau-Daten entwickelt werden, sollten zunächst die beabsichtigten Nutzungstrukturen in die Überlegungen eingehen. Nach diesen Nutzungsmöglichkeiten kann dann die Art der Clients ausgerichtet werden.

Wie schon in Kapitel 4.1 kurz beschrieben, soll der STLB-Bau-Auswahldialog zunächst als Client betrachtet werden, da er in der Nutzungskette zunächst an erster Stelle steht. Wie die Daten dann weiter verwendet werden sollen, bestimmt natürlich die Art dieses Clients wesentlich mit.

4.4.1 Clients für Windows-AVA-Systeme

Wie schon in der bestehenden Umsetzung sollen die STLB-Bau-Daten weiterhin Windows-AVA-Systemen zur Verfügung gestellt werden. Dies natürlich mit dem Unterschied, dass die Daten online gewonnen werden. Dies wird wohl auch wie bisher, die größte Nutzergruppe betreffen, da der hohe Investitions- und Einarbeitungsaufwand in diese AVA-Systeme dies rechtfertigt.

Abb. 10 zeigt das Modell eines solchen Clients und die Weiterleitung der Daten, wie es von den Entwicklern der Dynamischen Baudaten geplant ist. Der STLB-Bau-Auswahldialog soll als Komponentensoftware im Microsoft Komponentenmodel COM realisiert werden. Die darin enthaltene COM-Schnittstelle

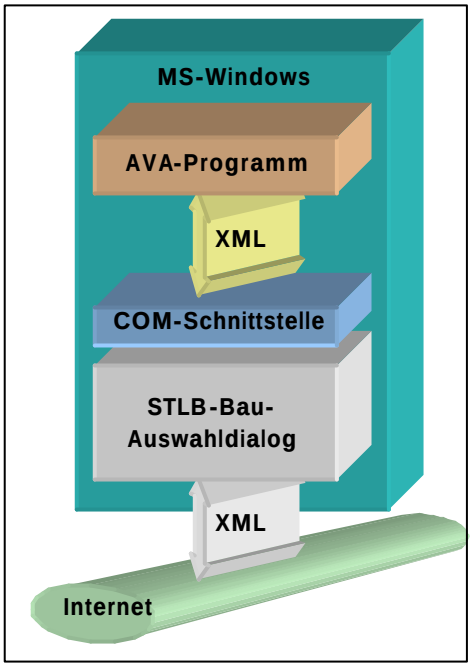

Abb. 10

Modell eines Clients für eine Windows-Plattform und Integration in bestehende AVA-Programme

stellt sicher, dass Windows-AVA-Systeme Zugriff auf diesen Auswahldialog besitzen.

Die Verbindung zum Server muss natürlich über die schon genannte XML-Schnittstelle erfolgen. Ebenso ist der weiterführende Datenaustausch zum Windows-AVA-Programm über XML geplant. Da alle Schnittstellen offengelegt sind, ist es Entwicklern natürlich auch möglich eigene Applikationen zu erstellen. Dies allerdings immer unter der Bedingung, dass es Windows-Applikationen sind. Um diesen Nachteil zu umgehen, liegt der erste Ansatzpunkt in der Schaffung eines plattformunabhängigen, möglichst unversell einsetzbaren Clients. Überlegungen hinsichtlich dieser Zielsetzung sollen im folgenden dargestellt werden

4.4.2 Universell einsetzare Clients

Verschiedene STLB-Bau-Auswahldialoge zu entwickeln, die auf mehreren heterogenen Client-Plattformen, wie beispielsweise Macintosh oder Unix laufen sollen, verursacht hohe Kosten und ist sehr arbeitsintensiv. Idalerweise hätte ein STLB-Bau-Client natürlich die Eigenschaft universell einsetzbar zu sein, also hinsichtlich der verwendeten Plattform und der Weitergabe der Daten an ein AVA-System flexibel zu sein.

Bedingung dafür ist eine plattformunabhängige Implementation und eine universell konfigurierbare Schnittstelle zu AVA-Systemen. Bevor allerdings konkrete Umsetzungsvorschläge in Kapitel 5 gemacht werden, sollten noch einige Überlegungen hinsichtlich der Weiternutzung der Daten aus dem Auswahldialog heraus gemacht werden.

4.4.2.2 Anbindung an clientseitig ablaufende AVA-Systeme

Als clientseitig ablaufende AVA-Systeme sollen im folgenden alle AVA-Systeme zusammengefasst werden, die auf der Arbeits-Plattform des Nutzers ablaufen.

Dazu gehören auch die schon im vorangegangenen Kapitel genannten Clients für Windows-AVA-Systeme, wie Systeme, die für andere Plattformen entwickelt wurden und auf der Nutzer-Plattform installiert sind.

Eingeschlossen in diesen Bereich sollen auch mögliche AVA-Systeme in Form eines Applets sein. Der Code dieses Applets wird

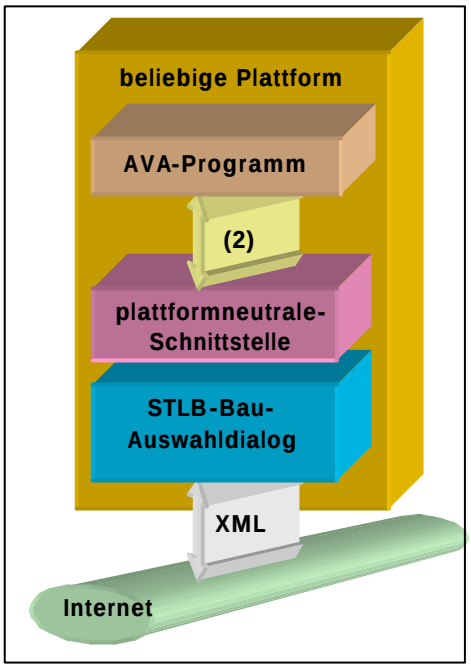

Abb. 11

Modell eines Clients für eine plattformunabhängige lokale ablaufende AVA-Applikation

zwar von einem Server geladen, läuft jedoch auch auf der Nutzer-Plattform.

Idealerweise sollte der STLB-Bau-Client allen diesen denkbaren Möglichkeiten von AVA-Systemen Daten zur Verfügung stellen können.

Im folgenden soll noch eine andere mögliche Architektur von AVA-Systemen gezeigt werden. Systeme, die auf einem Server ablaufen.

4.4.2.3 Anbindung an serverseitig ablaufende AVA-Systeme

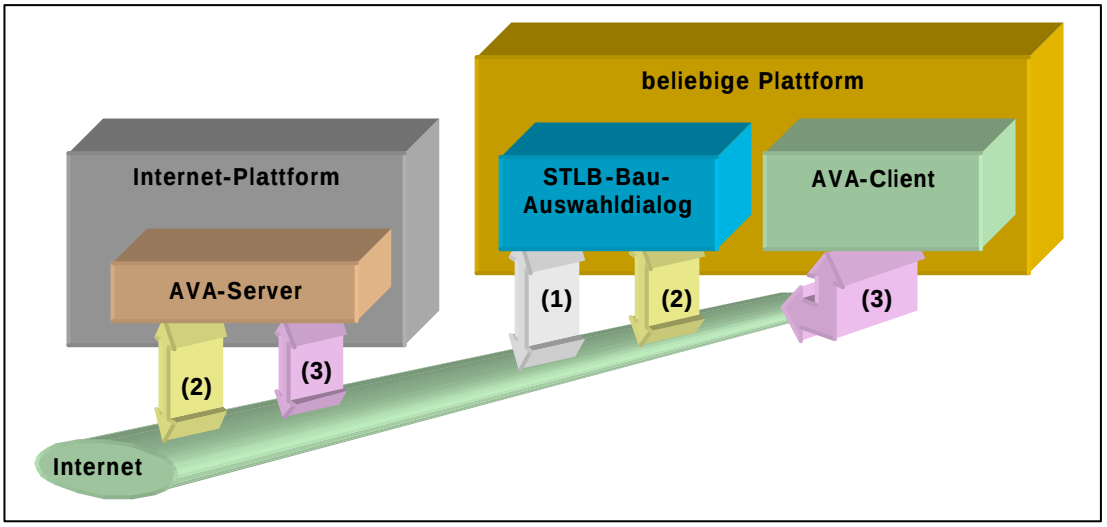

Abb. 12

Modell einers serverseitg ablaufenden AVA-Systems

Abb. 12 zeigt die Struktur eines denkbaren verteilten AVA-Systems. Ein Nutzer, dessen Arbeitsplattform beliebig ist, besitzt den Zugriff auf ein AVA-System, das auf einer Internetplattform läuft. Nach einem Anmeldungs- und Initialisierungsvorgang kann die AVA-Applikation vom Nutzer verwendet werden. Dabei ist ein gängiger Internetbrowser als Clientmaschine dieser AVA-Applikation vorgesehen. Dieser Client übernimmt die Darstellung und die Benutzerinteraktion, besitzt jedoch keinerlei Funktionalität hinsichtlich einer Verarbeitung von STLB-Bau-Daten.

Ein Ablaufszenario, das beim Erstellen eines Leistungsverzeichnisses mit diesem System vorstellbar ist, soll im folgenden geschildert werden.

Zum Anlegen einer Teilleistung soll das STLB-Bau benutzt werden. Dazu wird der entsprechende Auswahldialog geöffnet. Nach dem Zusammenstellen einer Teilleistung in diesem Auswahldialog soll die Teilleistung in das Leistungsverzeichnis übernommen werden. Dazu muss der Datensatz dieser Teilleistung vom AVA-Server der Internet-Plattform nach erfolgreicher Zusammenstellung gefordert werden. Nachdem dieser Datensatz über die Schnittstelle übertragen wurde, übernimmt dieser AVA-Server die weitere Arbeit und fügt diesen Datensatz in das zu bearbeitende Leistungsverzeichnis ein. Der Nutzer des Systems erwartet natürlich, dass das zu bearbeitende Leistungsverzeichnis immer auf dem aktuellen Stand angezeigt wird. Um dies zu verwirklichen, müssen also die aktuellen Daten an den Client zurückgeliefert

werden. Dieser visualisiert dann das aktuelle Leistungsverzeichnis und wartet auf weitere Befehle des Nutzers.

Diese Verteilung der Aufgaben mag zunächst etwas kompliziert erscheinen, bringt aber eine Reihe von Vorteilen mit sich, auf die später eingegangen werden soll.

Die Verteilung der Aufgaben an verschiedene Komponenten in diesem System hat natürlich zur Folge, dass Schnittstellen zwischen den kommunizierenden Komponenten geschaffen werden müssen. Diese notwendigen Schnittstellen sollen auch kurz näher beschrieben werden.

- Schnittstelle (1) stellt die Verbindung zwischen STLB-Bau-Server und –Client her.
- Schnittstelle (2) stellt die Verbindung zwischen STLB-Bau-Client und dem AVA-Server der Internetplattform her.
- Schnittstelle (3) stellt die Verbindung zwischen AVA-Server und AVA-Client her.

5. KONZEPTION EINES UNIVERSELLEN CLIENTS (AUSWAHLDIALOG)

In diesem Kapitel sollen mögliche Realisierungsformen eines plattformunabhängigen Clients für STLB-Bau-Daten, wie er in Kapitel 4.4.2 ansatzweise beschrieben ist, gesucht werden. Nachdem Anforderungen herausgestellt werden, sollen Realisierungsvorschläge gemacht werden. Dabei sollen die zur Verfügung stehenden Realisierungtechnologien und deren Konzepte im Vordergrund stehen.

5.1 VERWENDETE PROGRAMMIERSPRACHE

Um die Skalierbarkeit und insbesondere die Offenheit des STLB-Bau-Clients zu gewährleisten, sollte dieser in Java implementiert werden. Das Java-Konzept „Write Ones, Run Everywhere“ erscheint hinsichtlich einer plattformneutralen Implemementationsform am besten. Zudem besitzt den großen funktionellen Umfang einer objektorientierten Programmiersprache und ist.

5.2 KOMPONENTENBILDUNG

Eine Möglichkeit, die Offenheit eines Systems zu erhöhen, ist die Aufspaltung in Komponenten.

Schon das bestehende Programmmodul ist als Windows-DLL realisiert, und kann als relativ einfache Komponente angesehen werden. Für die Windows-spzifische Lösung ist COM vorgesehen.

Für die Implementierung einer plattformneutralen Komponente bietet sich das Java-Kompontenmodel der *Java-Beans* an. *Java-Beans* besitzen ähnliche Eigenschaften wie Java-Klassen, haben jedoch eine exaktere und für Programmierwerkzeuge verständlichere Schnittstellendefinition. Diese Schnittstellen, die den Charakter von *Beans* wesentlich bestimmen bezeichnet man als Features. Sie bestehen aus drei Teilen: *Properties*, *Methods* und *Events*. (in Abb.13 dargestellt)

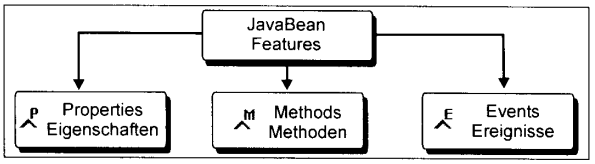

Abb. 13

Bestandteile der Java-Bean Schnittstelle

nach [6]

Properties stellen die öffentlichen Attribute oder Eigenschaften des Java-Beans dar. Über diese Properties sind *Beans* vom Programmierer anpassbar, ohne den Code zu ändern. *Methods* sind öffentliche Methoden, also Funktionen, die ein *Bean* ausführen kann. Als dritter Teil sind noch die Ereignisse zu nennen, die von einem *Bean* erzeugt werden können.

Diese drei Arten von Features sind im Interface der Klasse *BeanInfo* dokumentiert, das Interface *PropertyEditor* dient dem individuellen Anpassen von Java-Beans.

Das Komponentenmodell, das für Java gewählt wurde, ist zwar an diese Programmiersprache gebunden, aber dennoch offen. So besteht die Möglichkeit, fremde Komponenten wie *AktiveX-Controls* einbetten zu können. Empfehlenswert ist dies allerdings deshalb nicht, weil der große Vorteil der Plattformunabhängigkeit verloren geht. Aus diesem Grund sollten Java-Beans auch immer in Java programmiert werden.

Im Fall einer STLB-Bau-Client-Entwicklung sollte dieser Client also möglichst als Komponentensoftware gestaltet werden. Die Features dieser Komponente können im Moment noch nicht genau spezifiziert werden. Funktionell sind diese allerdings ähnlich der jetzigen Programmschnittstelle des STLB-Bau zu gestalten, jedoch bestehen Möglichkeiten, die Funktionalität auch in *Events* oder *Properties* zu realisieren.

Derartige Sofwarekomponenten benötigen sogenannte Container, in die sie eingebettet sind. Diese Aufgabe muss die spezielle Realisierungsvariante des AVA-Systems übernehmen. Diese Varianten wurden in Kapitel 4.4.2 beschrieben. Also entweder die auf der Nutzerplattform ablaufende AVA-Applikation bzw.-Applet oder der AVA-Client, der auf der Nutzerplattform läuft, wenn die eigentliche Applikation serverseitig gelagert ist.

5.3 Graphisches Userinterface

Eine Klassenbiblliothek zur Erstellung graphischer Bedieneroberflächen ist AWT (Abstract Window Toolkit). Hierin sind allerdings nur elementare Elemente einer Bedieneroberfläche enthalten.

Mit Java 2 kam eine weitere Oberflächenbibliothek neben AWT hinzu – Swing. Eine große Auswahl an weiter konfigurierbaren Standardelementen macht das Erstellen einer modernen und evtl. erweiterbaren Bedieneroberfläche möglich. So erscheint diese Klassenbibliothek besser für eine Umsetzung einer STLB-Bau-Client-Oberfläche geeignet.

5.4 Schnittstellen

5.4.1 Schnittstelle zum STLB-Bau-Server

5.4.1.1 Anforderungen an die Schnittstelle

Die eigentlichen Aufgaben des Auswahldialogs liegen wie schon genannt in der Datendarstellung und der Gewährleistung der Benutzerinteraktionen. In dieser Komponte ist allerdings keine Logik enthalten hinsichtlich der Zusammenstellung. Dies wird vom serverseitigen Datenzugriffsmodul erledigt.

Die Schnittstelle muss also Steuerbefehle und Daten liefern. Da sie von den Entwicklern der Dynamischen Baudaten spezifiziert und serverseitig

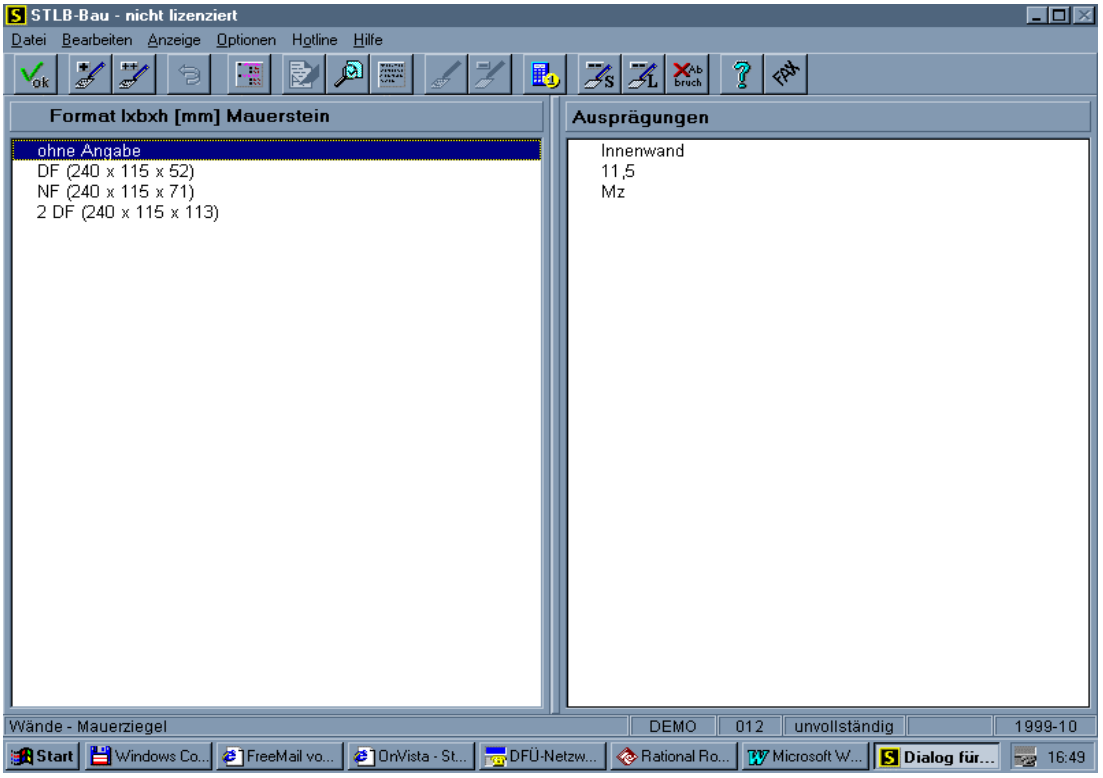

Abb.14

Programmodul STLB-Bau in seiner bestehenden Form

implementiert wird, muss sie von anderen Entwicklern als Standard angenommen werden. Konkrete Informationen über die Spezifikation dieser Schnittstelle sind allerdings momentan noch nicht verfügbar. Fest steht nur, dass XML die Basis bilden soll. In Abb.14 wird das bestehende Programmmodul STLB-Bau gezeigt. Von der Benutzerführung ist ein STLB-Bau-Client wahrscheinlich identisch zu gestalten. Dazu müssten die Datensätze in den beiden Anzeigefenstern stängig aktualisiert werden. Links werden auswählbare Beschreibungsmerkmale bzw. Ausprägungen, rechts die schon bestehenden Auswahlergebnisse dargestellt. Nach einer erfolgten Auswahl müsste das Ergebnis an den Server übertragen werden und die Fensterdaten aktualisiert werden.

5.4.1.2 Umsetzung

Für Java stehen eine Reihe von Klassenbibliotheken zur Verfügung. Eine XML-Klassen-Bibliothek ist die Sun Java Project X *Technology Release 2,* die frei verfügbar ist. Eine Klasse, welche die gegebenen Anforderungen realisieren könnte, ist die Klasse *XmlRpcClient*. Diese stellt Grundfunktionalitäten hinsichtlich XML Datenaustausch und entfernter Prozeduraufrufe mit dem Server zur Verfügung und könnte evtl. den o.g. Anforderungen angepasst werden. Als Übertragungsprotokoll wird das im Internet übliche HTTP/HTTPS benutzt.

XML über HTTP bzw. HTTPS zu übertragen ist nur eine von vielen Möglichkeiten diese Request/Response-Funktionalität zu realisieren. Sie besitzen allerdings eine Reihe von Vorteilen, die es zu beachten gibt. So

können Firewalls durchgangen werden. Firewalls sollen sensible Unternehmensdaten schützen. Solche Firewalls zu durchgehen, ist mit anderen Übertragungsprotokollen nur sehr schwer möglich. Ebenso ist das zur Verfügung stehende HTTPS-Protokoll das in Bezug auf die Sicherheit das am weitesten entwickelte Standard-Protokoll zur Sicheren Datenübertragung.

Allerdings zeichnen sich auch einige Nachteile ab. So ist es zu vermeiden, Daten in kleinen Paketen zu übertragen. Wegen der hohen Latenzzeiten, die im Internet auftreten können, kann es zu Geschwindigkeits-Problemen kommen. So ist es sehr zeitaufwendig, Daten Stück für Stück zu übertragen. Genau diese ist aber bei Benutzerinteraktion in einem STLB-Bau-Cleint nicht vermeidbar. Nach jedem Auswahlschritt müssen ständig Aktualisierungen erfolgen.

So scheint es zunächst vorteilhafter zu sein, zu warten bis die Spezifikation der Schnittstelle abgeschlossen ist, obwohl diesem Thema allerdings der Schwerpunkt der Arbeit zugewiesen werden sollte.

5.4.2. Schnittstellen zum AVA-System

Um die Anbindung an bestehende AVA-Programme, die nicht Windows-basiert sind zu gewährleisten, ist ein entsprechender STLB-Bau-Auswahldialog mit einer möglichst flexiblen Schnittstelle zum AVA-Programm bereitzustellen. Flexibel vor allem hinsichtlich der Realisierungsform des AVA-Systems. Mögliche Realisierungsformen sind in Kapitel 6 beschrieben.

5.5 Parallele Ausgaben-Abarbeitung

Um den Benutzer nicht auf eine Antwort warten zu lassen, sollte das Programm mit Hilfe von Threads parallele Schritte abarbeiten können. Dazu gehören mindestens die Bedienersteuerung um Bedienungsmöglichkeiten auch während stattfindenden Datenübertragungsprozessen möglich zu machen. Ebenso sind aber weitere Threads denkbar.

5.6 Absichern der Zuverlässigkeit

Hinsichtlich der Zuverlässigkeit des Clients sollte bei dem Entwurf und der Implementierung besondere Rücksicht den Standard gelten. Java-VM's bzw. die Java-Runtime-Umgebungen der bestehenden Browser sollten die implementierten Standards und Funktionalität auch unterstützen.

Sollte eine Java-Entwicklungsumgebung benutzt werden, sollte diese auch Unterstützung der beabsichtigten Standards bieten. So werden z.B. nicht alle Funktionen der aktuellen Version Java 2.0 von allen Java-Entwicklungsumgebungen unterstützt. Bestimmte herstellerspezifische Standards zu implementieren ist aus diesen Gründen nicht ratsam. Dies betrifft z.B. auf die Sicherheitsfunktionen zu, die in Kapitel 5.2.3 beschrieben werden.

6. KONZEPTION VON AVA-SYSTEMEN

6.1 ALLGEMEINES

Die Funktionalität eines AVA-Systems besteht im wesentlichen darin, Teilleistungen systematisch zu einem Leistungsverzeichnis zusammenzufügen. Zusätzlich sind jeder Teilleistung Preise zuzuorden. Die Herkunft dieser Preise ist je nach dem Stand des Bauprojekts und den Anforderungen des Bauunternehemens unterschiedlich. Individuelle Preisansätze, sowie Preise die aus angebotenen Bibliotheken stammen, sollten dafür verfügbar sein.

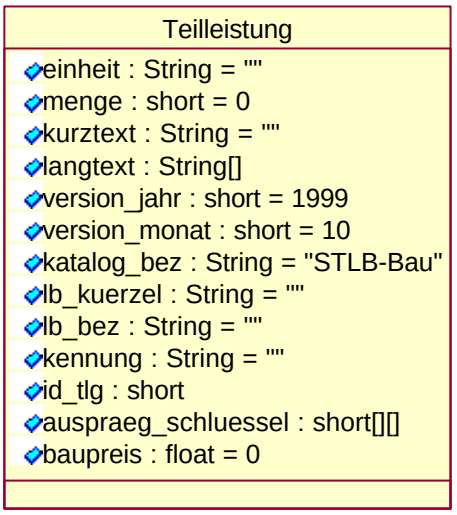

Abb. 15

UML-Darstellung der Klasse Teilleistung

Abb.15 zeigt die Klasse *Teilleistung*. Die Attribute dieser Klasse entsprechen im wesentlichen den schon bekannten Daten an der Programmschnittstelle des STLB-Bau.

Im AVA-Programm müssten solche Teilleistungen mit den entsprechenden Positionsnummern und Gliederungsstrukturen zu einem Leistungsverzeichnis zusammengesetzt werden. Dies könnte mit Hilfe von listen- oder feldartigen Strukturen geschehen.

6.2 KONZEPTION EINES CLIENTSEITIG ABLAUFENDEN AVA-SYSTEMS

6.2.1 Applikation oder Applet

Java-Applikationen entsprechen herkömmlichen Anwendungen. Sie stellen jedoch keine ausführbaren Programme dar, sondern müssen in einem Laufzeitsystem gestartet werden. Dies besteht aus einem Java-Interpreter, der zusammen mit diversen Bibliotheken ausgeliefert wird und die *Java-VM* (*Java Virtual Machine)* darstellt.

Applets stellen eine Variante von Java-Programmen dar. Sie können innerhalb eines Java-fähigen Web-Browsers ablaufen. Dieser hat dann das Laufzeitsystem bereits integriert. Der Browser lädt das in eine HTML-Datei eingebettete Applet und besitzt dann die weitere Kontrolle für den Programmablauf. Für Applets gelten aus Sicherheitsgründen eine Reihe von Einschränkungen, auf die in Kapitel 5.1.3 näher eingegangen wird.

Der Vorteil eines solchen Applets besteht darin, dass dem Benutzer immer eine aktuelle, funktionsfähige Programmversion zur Verfügung steht. Er muss sich also keine Gedanken um eine lokale Installation machen. Nachteile bestehen zum einen darin, dass dieses Applet bei jedem Start erneut vom Web-Server geladen werden muss, was bei großen Programmen und langsamen Internet-Verbindungen erheblichen Zeitaufwand bedeuten kann. Zum anderen müssen die schon angesprochen Sicherheitsbeschränkungen aufgehoben werden.

Es ist allerdings durchaus möglich, Java-Programme zu schreiben, die gleichzeitig Applikationen und Applets sind.

Bei ersten Implementierungsversuchen, hinsichtlich der Simulierung eines Datenaustauschs zwischen einem Java-Applet und STLB-Bau-Client entstand ein Applet, dass zur Anzeige von übergebenen Daten fähig ist. Unter der URL „http://www.uni-weimar.de/~heinszma/index.html“ ist dieses aufrufbar und auch der Quellcode zu finden. Zu Testzwecken hinsichtlich einer Implementierung des Datenaustausches könnte dies evtl. noch nützlich sein.

6.2.2 Sicherheit

Da Java-Applets in HTML-Dokumente gebettete Programme sind, die praktisch von „außen“ auf den lokalen Rechner geladen werden, gelten diese von vorn herein als unsicher. Deshalb gelten zumindest für diese unsignierten Applets eine Reihe von Funktionseinschränkungen, die in Abb.16 aufgezählt werden.

- Das Applet darf keine Dateien lesen oder Schreiben, löschen, umbenennen oder Informationen über diese abfragen
- Es dürfen keine Verzeichnisse angelegt, sowie Informationen über Verzeichnisse oder deren Attribute ermittelt oder verändert werden
- Die Weitergabe von Informationen über das System und den Benutzer, zum Beispiel Rechner oder Benutzernamen, sind verboten.
- Das Applet darf außer der bestehenden Verbindung zum Webserver, von dem das Applet geladen wurde, keine neuen Netzwerkverbindungen aufbauen.
- Es dürfen keine lokalen Programme gestartet (wie format.com) und keine nativen Bibliotheken (z.B. DLLs) eingebunden werden.
- Es dürfen keine „fremden“ Threads beeinflusst werden. Das sind alle Threads, die nicht der Thread-Gruppe des Applets angehören.
- Die Klassen der Java-Systemklassenbibliothek (Core-API). Java.*) dürfen nicht durch andere Klassen oder Versionen überschrieben werden. Ansonsten ließen sich obige Restriktionen durch Überschreiben der Security-Klassen leicht ausschalten.
- Die Ausgabe auf einem Drucker ist ebenfalls nicht gestattet.

Abb.16

Restriktionen unsignierter Applets

nach [6]

Wenn man daran denkt, ein Applet mit auch nur annähernder AVA-Funktionalität zu implementieren, ist man allerdings auf einen Großteil der Funktionen angewiesen, die hier eingeschränkt werden.

So müsste zumindest auf lokale Dateien zugegriffen werden können oder eine Netzwerkverbindung aufgebaut werden können, um einmal erzeugte Leistungsverzeichnisse ablegen zu können oder zu exportieren.

Ziel müsste es also sein, „sichere“ Applets oder auch signierte Applets zu erzeugen. Dieses Verfahren erscheint zunächst sehr aufwendig, denn um diese digitale Unterschrift zu leisten, muss sich der Signierer bei einer authorisierten Zertifizierungsbehörde registrieren lassen, die seine Identität bestätigt. Zudem gibt es in der Praxis Probleme mit eigenen Sicherheitsmodellen der Hersteller von Web-Browsern. Ein weiteres Problem bei Java 1.1 ist, dass mit dem Erwerb eines Zertifikates automatisch alle Sicherheitsbeschränkungen aufgehoben werden.

Insbesondere in Java 2, gibt es Möglichkeiten, Teile dieser Restriktionen aufzuheben. Für die einzelnen Ressource-Zugriffe gibt es hier spezielle Zugriffsrechte. In einer abgesicherten Laufzeitumgebung stellt der auszuführende Code Anfragen für bestimmte Zugriffe. Anhand der Quelle des Codes und den mitgeführten Zertifikaten wird durch Vergleich mit den lokalen Zugriffsrechten entschieden, ob eine Anfrage erlaubt ist.

6.3 Konzeption eines serverseitig ablaufenden AVA-Systems

6.3.1 Allgemeines

Die zu damaliger Zeit visionäre Philosophie „The Network is the Computer“ der Firma Sun Microsystems ist momentan dabei, sich langsam in Realität zu verwandeln. Der wichtigste Bestandteil dieser Philophie ist die Verlagerung der Funktionalität auf die Seite von Servern. Bisherige erschlossene Anwendungsbereiche bestehen vor allem in den Bereichen von Online-Banking- und Online-Shoping-Lösungen.

Es bestehen verschiedene Techniken und Programmiersprachen für serverseitige Anwendungen. Allen gemeinsam ist jedoch, dass sie Dokumente erzeugen, welche von einem Server zu einem Client gesandt werden. Dieses Grundprinzip wird als *Server Side Scripting* bezeichnet.

Clientseitig können die von diesen Programmen generierten Dokumente mit einem beliebigen Browser angezeigt werden. Diese Dokumente dienen allerdings nicht nur zur Anzeige sodern enthalten auch Interaktionsmöglichkeiten, stellen also praktisch die Grafische Benutzeroberfläche der serverseitig ablaufenden Anwendung dar. Im Allgemeinen handelt es sich bei diesen Dokumenten um HTML-Seiten.

6.3.2 Servlets

Im Gegensatz zum Applet, läuft der abzuarbeitende Java-Code hier auf der (sicheren) Serverseite ab. Die in Kapitel 6.2.2 angesprochenen Sicherheitsbeschränkungen für Applets spielen hier also keine Rolle.

In der Praxis stellt sich das Konzept folgendermaßen dar: Der Client - in der Regel ein Standard-Webbrowser - stellt eine Anfrage an einen Webserver. Dieser startet das angesprochene Java-Servlet und baut zum Beispiel eine Verbindung zu einer Datenbank auf. Innerhalb des Servlets wird die Anfrage verarbeitet und anschließend das Ergebnis in Form von generierten HTML-Ausgaben an den Webserver und somit an den Web-Browser zurückgesendet.

Um mit Servlets zu arbeiten muss der Web-Server eine entsprechende Unterstützung bieten. In Version 2 des JDK besteht die Möglichkeit, Servlets zu programmieren. Im Java-Servlet-API unter dem Namen *Java Servlet Development Kit (JSDK*) werden entsprechende Basis-Klassen bereitgestellt. Besonders die Klasse *HttpServlet* wäre hinsichtlich einer serverseitigen AVA-Applikation interessant.

Servlets können auf ganz verschiedene Art und Weise aufgerufen werden, dazu gehören u.a. der direkte Aufruf über eine URL. Die Aufgabe eines AVA-Servlets wäre es also Anfragen des Clients zu empfangen, diese zu bearbeiten und die Ergebnisse in Form einer dynamisch generierten HTML-Seite an den Client zu senden.

Ein auftretender Nachteil ist hierbei, dass im Code eines AVA-Servlets sowohl die AVA-Funktionalität als auch Funktionalität hinsichtlich des generierenden HTML-Codes enthalten sein muss. Angaben zum Design der Web-Seiten befinden sich deshalb verstreut im Code des Servlets. Änderungen des Designs erfordern jedes Mal eine Neuübersetzung. Noch schwerwiegender ist der Umstand, dass die Erstellung von Web-Anwendungen mittels Servlets gute Java-Kenntnisse voraussetzt. Erst eine weitgehende Trennung von Oberflächengestaltung und Programmierung ermöglicht eine produktive Zusammenarbeit von Designern und Programmierern.

Die in letzter Zeit stark vorangetriebene Entwicklung von server-seitig ablaufenden Applikationen brachte u.a. die Technologie der *Java-Server-Pages* hervor, die als Erweiterung der JavaServlet API eingeordnet werden kann.

6.3.3 Java Server-Pages (JSP)

Die erste JSP Spezifikation wurde von *Sun Microsystems* im Juni 1999 veröffentlicht. Die erste Implementation von Sun, in der die JSP Spezifikation 1.0.1 und die Servlet API 2.1 implementiert wurden, war das Java Server Web Development Kit (JSWDK). Um verschiedenen Sicherheitsanforderungen

gerecht zu werden, stehen ebenfalls Erweiterungen zur Verfügung. Für *JSP* gibt es die Erweiterung JSSE (*Java Secure Socket Extension*) von Sun.

Die Java Server Pages (JSP)-Technologie setzt genau an der Stelle der Trennung von Design und Programmierung an. Bei Servlets wird HTML-Code in den Programmcode eingebettet, JSP geht genau den umgekehrten Weg: Programmcode wird in HTML-Dokumente eingebettet. Der Code wird dabei weitgehend in Java Beans und benutzerdefinierten Tags versteckt. Dies ermöglicht die getrennte Entwicklung von Oberflächengestaltung und Programmlogik.

Diese Trennung eröffnet für den Benutzer vielfältige Möglichkeit. So können auf relativ einfache Art personalisierte Oberflächen angeboten werden. Besitzt der Server Informationen hinsichtlich des Benutzers, so können z.B. die Sprache, tageszeit- oder länderspezifische Inhalte flexibel an Benutzerbedüfnisse angepasst werden. Eine übliche Art, dem Server solche Client-Informationen bekannt zu machen sind Cookies, die nach einem Anmeldevorgang auf der Clientplattform angelegt werden. Für speziellere Informationen ist ein Eintrag in einer Nutzerdatenbank auf dem Server erforderlich.

Für die Entwickler eines solchen Systems ergeben sich ebenso eine Reihe von Möglichkeiten. So ist eine feine Modularisierung möglich, welche die Beherrschbarkeit von sehr komplexen Systemen erhöht. In Abb.17 wird ein Modell gezeigt nachdem diese Modularisierung anufgebaut werden kann.

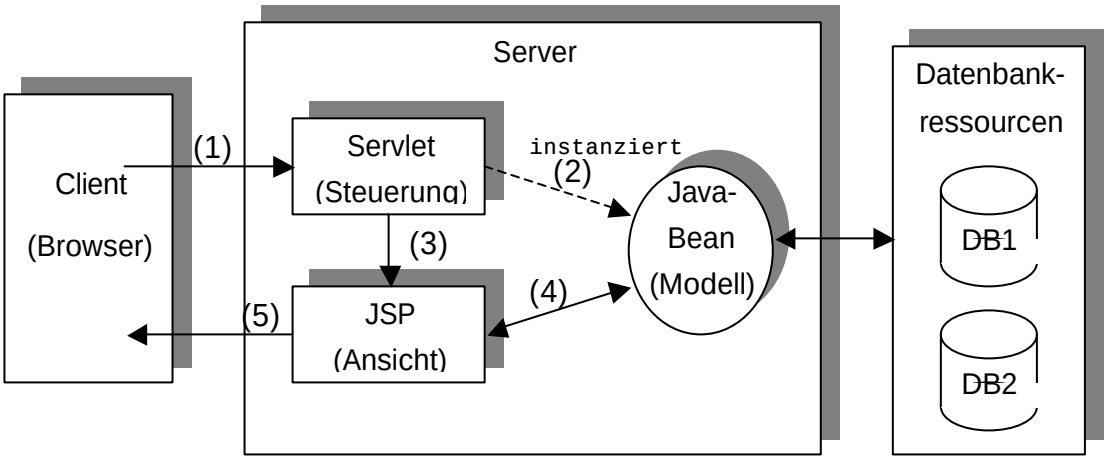

Abb. 17

Model-View-Controller (MVC) Paradigma

Dem Modell entsprechen die Java-Beans mit ihren Attributen und den dazugehörigen schreibenden und lesenden Methoden. Die Beans werden hier auch als Anwendungslogik bezeichnet.

Der JSP-Seite fällt die Aufgabe der Sicht zu. Durch HTML- und XML-Tags, sowie durch eventuelle Skriptteile wird die Optik der Seite, die beim Client ankommt, generiert. Der JSP-Seite kommt hierbei keine datenverarbeitende Aufgabe zu, sondern nur die der Präsentation.

Der Steuerung entspricht in der JSP-Praxis das Servlet. Das Servlet nimmt die Anfrage des Benutzers entgegen, um dann die benötigten Beans zu

instanziieren bzw. zu lokalisieren. Eine JSP-Seite wird aufgerufen, welche sich mit den nötigen Bean-Daten versorgt, um daraufhin eine der Anfrage entsprechende HTML-Seite an den Client zu schicken.

Auf den konkreten Fall einer AVA-Applikation bezogen wurde sich dies folgendemaßen darstellen:

Der Client initiiert einen Befehl, z.B. das Einfügen einer neuen Teilleistung in ein Leistungsverzeichnis. Der STLB-Bau-Client wird geöffnet und die Daten der Teilleistung werden zusammengestellt. Bis zu diesem Punkt läuft alles noch clientseitig ab.

Nach dem Übernehmen der Daten an den Auswahldialog müssen diese mit dem entsprechenden Kommando, z.B. Einfügen an den AVA-Server, bzw. an das Servlet gegeben werden. An ein Java-Bean, dessen funktioneller Bereich es ist, Leistungsverzeichnisse zu strukturieren, könnten diese Informationen dann weiter gegeben werden. Ein anderes Bean, dass für Datenbankzugriffe zuständig ist, könnte die Aktualisierung des LV's das in der Datenbank abgelegt ist, übernehmen. Eine denkbare Aufgabe eines Beans könnte die Generierung von Preisansätzen, der Teilleistung sein.

Nach dem Anlegen einer JSP-Seite könnte ein darin eingebetteter weiterer Bean die Daten des aktualisierten LV's auslesen und in HTML konvertieren.

Nachdem alles an den Browser des Benutzers zurückgeschickt und dort angezeigt wurde, hat der Benutzer das aktualierte Leistungsverzeichnis auf dem Monitor und kann weitere Bearbeitungsschritte vornehmen.

Ein Entwurf der Ablauflogik eines solchen Systems ist im Anhang zu finden.

Durch die Trennung der drei Bereiche erreicht man für die Entwicklung Übersichtlichkeit, so dass man die einzelnen Aufgaben auch einzelnen Entwicklungsgruppen unterstellen kann, die dann zu großen Teilen getrennt voneinander arbeiten können. Dies beschleunigt die Entwicklungsdauer von Applikationen, wobei deutlich Modularität und Wiederverwendbarkeit betont werden.

Die Projektplanung einer solchen Applikations-Entwicklung bedeutet allerdings einigen Aufwand, da man die Bereiche klar voneinander trennen muss, um Sie am Ende wieder zusammen zu setzten.

7. ZIEL EINER OPTIMALEN LÖSUNG

Vor allem hinsichtlich der Zielsetzung einer Optimierung der Geschäftsprozesse bei Bauprojekten ist wohl die Lösung einer serverseitigen AVA-Applikation am aussichtsreichsten. Besonders wenn man bedenkt, dass diese AVA-Applikation in eine Internet-Plattform integriert werden könnte, auf der die am jeweiligen Bauprojekt Beteiligten zusammengeführt werden könnten.

Der STLB-Bau Client der die Versorgung der AVA-Applikation mit Daten aus dem Standardleistungsbuch übernimmt, könnte dann auf der Seite des Clients der AVA-Applikation ablaufen. Also idealerweise aus dem Browser heraus, als Applet gestartet werden. Es ist nur sicher zu stellen, dass Applet in dem der Auswahldialog enthalten ist, im Cache des Browsers abgelagert bleibt und nicht ständig neu geladen werden muss.

Wichtig um eine Akzeptanz bei den Nutzern zu finden, ist allerdings eine ausreichende Bedienerfreundlichkeit. Dies nicht nur hinsichtlich einer klaren einfachen Struktur der Bedienerführung und Benutzeroberfläche, sondern auch in der Arbeitsgeschwindigkeit des gesamten Systems. Wenn man die Vielzahl der Schnittstellen unter den einzelnen Komponenten betrachtet, so sind an diesem Punkt sicherlich einige Herausforderungen zu bewältigen.

LITERATURVERZEICHNIS

[1] GAEB in Verbindung mit der Dr. Schiller & Partner GmbH, StLB-Bau – Programmschnittstelle, April 1999, DIN e.V.

[2] Dr. Schiller & Partner GmbH – Dynamische Baudaten – Dresden, Kompendium Dynamische Baudaten, 1999

[3] A. Austermann, J. Gallenbacher, Ch. Lange, M. Spörl: Java 2 mit Methode. C&L Computer und Literaturverlag, 1999.

[4] William J. Pardi: XML in Action, Microsoft Press Deutschland, Unterschleißheim, 1999.

[5] Simon St. Laurent: XML – A PRIMER, IDG Books Worldwide, Foster City, CA, USA, 1998.

[6] Florian Hawlitzek: Java-Programmierung mit IBM Visual Age, Addison Wesley Longman Verlag, 1999

[7] GEAB – Gemeinsamer Ausschuß Elektronik im Bauwesen: Regelungen für Informationen im Bauvertrag (Aufbau Leistungsverzeichnis; GEAB-Datenaustausch 2000), BEUTH VERLAG GMBH, Version 1.0, Ausgabe Nov. 1999

Verwendete Software:

Zur Erstellung des Applets wurde die Java Entwicklungsumgebung *IBM-Visual Age for Java 3.0*