Universität Kaiserslautern
Fachbereich Elektrotechnik
Lehrstuhl für Digitale Systeme
Prof. Dr.–Ing. S. Wendt

Diplomarbeit

Konzept– und Prototypentwicklung einer Komponente zur zentralen Abwicklungssteuerung eines „Corporate Memory“–Systems

Markus Cherdron
März 1998

Betreuer: Prof. Dr.–Ing. S. Wendt

Bearbeiter: Markus Cherdron
Sudetenstr. 3
69190 Walldorf

Erklärung:

Hiermit erkläre ich, die vorliegende Diplomarbeit unter Verwendung der angegebenen Quellen und ohne fremde Hilfe angefertigt zu haben.

Walldorf, den 30. März 1998

Markus Cherdron

Danksagung

An dieser Stelle möchte ich allen danken, die mir nicht nur in fachlichen Fragen stets zur Seite gestanden haben.

Ein riesiges „Danke Schön“ gilt meiner Freundin Anke. Wer weiß, ob ich es ohne sie überhaupt zu Ende gebracht hätte. Auch in schweren Zeiten hat sie mich immer wieder unterstützt.

Meine Eltern gehören zu den wenigen Menschen, die in diesem Moment nachvollziehen können, wie glücklich ich mich fühle. Helga und Philipp verdanke ich sehr viel. Ohne ihre ständige aufopferungsvolle Hilfsbereitschaft, wäre vieles nicht möglich gewesen. Danke.

Mein Dank gilt Herrn Prof. Dr.–Ing. Siegfried Wendt, der mir durch seine Kompetenz und Menschlichkeit immer ein großes Vorbild sein wird. Ich empfand es stets als eine Ehre mit ihm und für ihn arbeiten zu dürfen. Die mit seiner Hilfe gewonnenen Erkenntnisse werden mich auf meinem weiteren Lebensweg stets begleiten.

Diese Diplomarbeit entstand im Software Engineering Labor der Firma SAP AG in Walldorf im Rahmen des Projekts TimeLess. Mein Dank gilt meinen unmittelbaren TimeLess Kollegen Christian Brand, Jürgen Langhauser und Stefan Steißlinger, die mit mir insbesondere während der Implementierungsphase durch Dick und Dünn gegangen sind.

Aber auch den anderen Kollegen hier im SEL gebührt mein Dank. Dr. Frank Gales, Bernhard Gröne, Eberhard Iglhaut, Andreas Knöpfel, Johannes Otto und Oliver Schmidt waren immer für anregende Diskussionen zu haben. Die Freundschaftlichkeit und Toleranz all meiner Kollegen gepaart mit dem menschlichen Führungsstil unseres Laborleiters Prof. Siegfried Wendt trugen zu der hervorragenden Atmosphäre des Labors bei.

Markus Cherdron

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1 Einleitung

Die Produktivität moderner Unternehmen hängt immer mehr von der Fähigkeit ab, Information optimal verwalten und nutzen zu können. Die Frage, „Wie bekommen meine Mitarbeiter möglichst schnell die Informationen, die sie benötigen, in einer für sie verständlichen Form?“ wird von Managern informationsproduzierender Unternehmen immer häufiger gestellt. Insbesondere die Unternehmen, die fast ausschließlich Information produzieren, sind gezwungen diese Frage befriedigend zu beantworten.

Ein Vertreter dieser Klasse von Unternehmen ist das Softwarehaus. In dieser Branche zeichnet sich immer mehr ab, daß die eigentlichen Informationsträger die Mitarbeiter sind, nicht die CDs, Disketten oder sonstigen Medien, auf denen Software gespeichert und als Produkt ausgeliefert wird. Das Wissen in den Köpfen der Mitarbeiter kann man daher als das Gedächtnis des Unternehmens oder auch „Corporate Memory“ bezeichnen.

Verliert das Unternehmen seine erfahrenen Mitarbeiter, ist das vergleichbar mit dem Verlust eines Teils des Gedächtnisses und wirkt wie eine Amnesie. Im Extremfall kann das Unternehmen dadurch vollständig gelähmt werden. Daher muß das Unternehmen an einer optimalen „Gedächtnisverwaltung“ interessiert sein. Dazu zählt auf der einen Seite sicherlich eine optimale Personalverwaltung. Noch wichtiger ist es, das Wissen der Mitarbeiter in einer informationellen Form, für das Unternehmen jederzeit zugänglich, abzulegen.

Die Entwicklung eines solchen Informationslogistiksystems stellt allerdings eine große Herausforderung dar. Wie ist das Wissen der Mitarbeiter überhaupt faßbar? Wie muß das erfaßte Wissen abgelegt werden, damit es jedem zugänglich ist, der es benötigt? Diese Fragen und andere beschäftigen seit Anfang 1997 die Mitarbeiter des TimeLess Projekts im Software Engineering Labor der Firma SAP AG unter der Leitung von Herrn Prof. Dr.–Ing S. Wendt. Die Bedeutung des Akronyms „TimeLess“ ist in Abbildung 1–1 dargestellt. Es handelt sich dabei um ein Softwaresystem zur Verwaltung sämtlicher, in einem Unternehmen anfallender Information, ähnlich einer virtuellen Bibliothek.

TimeLess steht allerdings auch für eine Philosophie. Klare Strukturen, Natürlichkeit sowie Benutzerorientierung sind einige der Begriffe, die TimeLess prägen. Bereits in den vorangegangenen Diplomarbeiten [Brand97] und [Langhauser97] wurden einige Konzepte erarbeitet,

die aufzeigen, wie diese Philosophie gewinnbringend in einem Produkt umgesetzt werden kann.

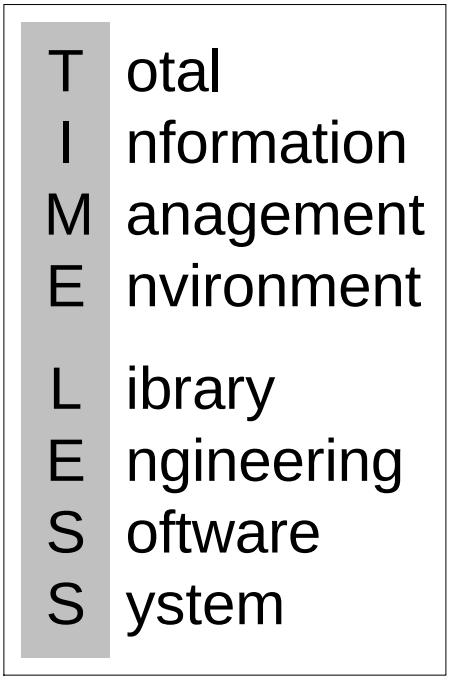

Abbildung 1–1: Das TimeLess–Akronym

Das Ziel dieser Arbeit ist es, den bisher eingeschlagen Weg fortzusetzen, um in die Praxis umsetzbare Softwarekonzepte zu erarbeiten, welche die Handschrift der TimeLess Philosophie tragen.

Das Nahziel des TimeLess Projekts ist die Entwicklung eines Abteilungsprodukts, welches im Rahmen dieser Entwicklungsbteilung von Mitarbeitern des Labors unter wohlwollenden Aspekten eingesetzt werden soll. Am Abteilungsprodukt kann untersucht werden, mit welcher Qualität die erarbeiteten Konzepte umgesetzt werden konnten. Die daraus gewonnenen Erkenntnisse dienen als Grundlage einer anschließenden Weiterentwicklungsphase im Rahmen des erklärten Fernziels des TimeLess-Projekts: Die Konstruktion eines marktfähigen TimeLess–Produkts.

Das Anwendungsfeld von TimeLess, nämlich die Informationslogistik, bietet sich darüber hinaus an, um Anforderungen an Architekturkonzepte großer Softwaresysteme zu untersuchen. Durch die Forschungsaktivitäten des Software Engineering Labors im Bereich der Architekturfindung durch Anwendungsanalyse werden hier enorme Synergieeffekte durch die Entwicklung von TimeLess erwartet. Auf der Suche nach einem geeigneten Architekturkonzept für große Softwaresysteme, können die aus TimeLess gewonnenen Erkenntnisse entscheidende Meilensteine darstellen.

1.1 Inhalt dieser Arbeit

Im Rahmen dieser Arbeit sollen einige Themen angesprochen werden, die insbesondere den Aspekt der Architekturfindung behandeln. Um dem Nahziel des Abteilungsprodukts näher zu kommen, wurde ein allgemeines Architekturkonzept entwickelt, anhand dessen die Entwicklung eines Prototyps vollzogen wird. Ausführungen zu diesem Architekturkonzept stellen die Hauptaufgabe dieser Arbeit dar.

Um das in Kapitel 5 ausgearbeitete Architekturkonzept besser verstehen zu können, wird in Kapitel 4 ein Unternehmensmodell eingeführt, welches durch seine Natürlichkeit und Praxisbezug leicht nachvollziehbar sein sollte. Das darin beschriebene fiktive Unternehmen (Informationsversandhaus) zeigt erstaunlich viele Parallelen zum Architekturmodell in TimeLess auf. Diese Parallelen verdeutlichen, daß es möglich ist, die sehr komplexen Sachverhalte großer Systeme, durch einen natürlichen Bezug zu bekannten Umgebungen, besser darzustellen und zu vermitteln. Dieser, in der Informatik nur zu selten anzutreffender Ansatz könnte zu sehr viel mehr Klarheit und Offenheit beim Systementwurf führen.

Die verschiedenen Themen, die im Rahmen des Architekturkonzepts behandelt werden, lassen sich aus den in Kapitel 3 aufgestellten Anforderungen ableiten. Dabei werden sowohl spezifische Anforderungen aus TimeLess behandelt, als auch nicht–funktionale Anforderungen, die Softwaresysteme dieser Klasse allgemein betreffen.

Zu Beginn dieser Arbeit (Kapitel 2), wird das im TimeLess–Labor angewandte Entwicklungsprozeßmodell näher erläutert. Aus diesem Prozeßmodell gehen Erkenntnisse hervor, die in einer später folgenden Produktentwicklungsphase von sehr großer Bedeutung sein können. In einer kleinen Entwicklergruppe ist die explizite Vorgabe eines Entwicklungsprozeßmodells insbesondere während der Vorentwicklungsphase nicht unbedingt von großer Bedeutung. Sobald die Anzahl der Entwickler jedoch ansteigt und der Fortschritt des Projekts zu jedem Zeitpunkt bekannt sein muß, damit die gesetzten Zwischenziele termin– und qualitätsgerecht erreicht werden können, spielt das explizite Projektmanagement eine große Rolle.

Kapitel 2 Der Entwicklungsprozeß

Das langfristige Ziel des TimeLess–Projekts ist es, ein produktiv einsetzbares System zur Verwaltung und Aufbewahrung von Unternehmenswissen (Corporate Memory) zu entwikkeln. Bedingt durch die Komplexität eines solchen Systems ist eine genaue Kenntniss der Entwicklungsprozesse notwendig. Um möglichst schnell die Zweckmäßigkeit eines solchen Systems evaluieren zu können, wurde als Prozeßmodell das sogenannte Prototyping[1] eingesetzt. Hierbei werden ein oder auch mehrere Prototypen erstellt, die dazu beitragen können folgende Probleme zu vermeiden:

- Fehlende oder falsche Anforderungen werden früh aufgedeckt.
- Die Entwickler finden schnell zu einer gemeinsamen Basis, welche die Kommunikation untereinander erleichtert und Verständnisprobleme minimiert.
- Eine Evaluation der technischen Realisierbarkeit eines den Anforderungen entsprechenden Produkts wird sehr früh und in der Regel auch relativ kostengünstig ermöglicht.
- Vor– und Nachteile eingesetzter Technologien können dargelegt werden. Das hat den zusätzlichen positiven Nebeneffekt, daß ein konsequentes und effektives Erlernen der Technologien durch die Entwickler ermöglicht wird. So ist es möglich, daß auch technologisch weniger erfahrene Entwickler durch den hohen Lerneffekt schneller in das Projekt integriert werden können.
- Ein Prototyp kann häufig zur Entscheidungsfindung beitragen. Sowohl technische und architekturelle Entscheidungen als auch solche, die die Projektplanung an sich betreffen, können bereits im Vorfeld detaillierter begründet werden.

Beim Prototyping gibt es viele unterschiedliche Ansätze, die bereits erfolgreich in anderen Projekten eingesetzt wurden. An dieser Stelle sollen jedoch nur zwei Unterklassen der Proto-

[1] Prototyping sowie weitere Software–Prozeßmodelle findet man in [Sommerville96]

typerstellung unterschieden werden: Der „Wegwerfprototyp“ sowie der „inkrementelle Prototyp“.

2.1 Wegwerfprototyp (Throw–away Prototyping)

Ein Wegwerfprototyp dient der genaueren Findung und Festlegung von konkreten Anforderungen, ohne die eine klare und verständliche Systemspezifikation sehr schwer und in der Regel nur abstrakt möglich wäre. Nach deren Beurteilung werden Prototypen dieser Klasse weggeworfen bzw. zur Verifikation der im getrennt zu entwickelnden Produkt enthaltenen Eigenschaften eingesetzt. Der in [Brand97] entwickelte Oberflächenprototyp gehört zur Klasse der Wegwerfprototypen. Die Erkenntnisse, die aus ihm gewonnen wurden, ermöglichen eine sehr klare Formulierung von Anforderungen und bilden auch die sprachliche Grundlage, die benötigt wird, um möglichst effektiv über TimeLess und deren Funktionalität zu reden.

2.2 Inkrementeller Prototyp

Im Gegensatz zum Wegwerfprototyp wird beim inkrementellen Prototyp das Ergebnis eines Entwicklungsschrittes nicht weggeworfen, sondern als Eingangsprodukt der nächsten Entwicklungsstufe benutzt. Das Gesamtziel wird in kleinere Einheiten partitioniert, die durch Integration mit vorhergehenden Einheiten etappenweise erweitert und schließlich zum Endprodukt führen.

Abbildung 2–1 verdeutlicht den Entwicklungsprozeß beim inkrementellen Prototyping. Man erkennt, daß sowohl die Formulierung der Anforderungen als auch der Entwurf des für das Gesamtsystem geltenden Architekturkonzepts bereits vor Beginn des ersten Entwicklungsinkrements stattfindet. Dies ist ein wichtiger Bestandteil dieser Art des Prototyping, da somit erreicht werden kann, daß spätere Inkremente nicht zu Änderungen früherer Entwicklungseinheiten führen müssen. Dies ist häufig die Konsequenz eines sich stets ändernden, neuen Gegebenheiten angepaßten Architekturkonzepts.

Die Konsequenz daraus besagt allerdings, daß die für einen solchen Prototyp zugrundeliegende Architektur sehr solide und für das Anwendungsgebiet geeignet sein muß. Dies erfordert große Anstrengungen bereits vor den eigentlichen Entwicklungsphasen, da viele Entwicklungsszenarien im Rahmen des erarbeiteten Architekturkonzepts durchgespielt werden müssen, um deren Eignung verifizieren zu können. Dies ist in der Regel nur mit einem hohen Grad an Abstraktionen zu erreichen und stellt somit einen der anspruchsvollsten Schritte der Gesamtentwicklung dar.

Im Idealfall würde man das System basierend auf einer Universalarchitektur[1] entwickeln, die für dieses Anwendungsgebiet geeignet scheint. Allerdings ist dies oft nur eingeschränkt möglich, da eine Evaluation solcher Universalarchitekturen in vielen Fällen sehr zeitaufwendig ist und deren Einsatz meist zu architekturellen Kompromissen zwingt. Dies liegt allerdings auch daran, daß für viele Anwendungsgebiete bisher noch keine kommerziellen Produkte entwik-

[1] vielfach spricht man hier auch von Frameworks

kelt werden konnten, die den oftmals sehr hohen Ansprüchen an die Architektur eines zu entwickelnden Softwaresystems genügen.

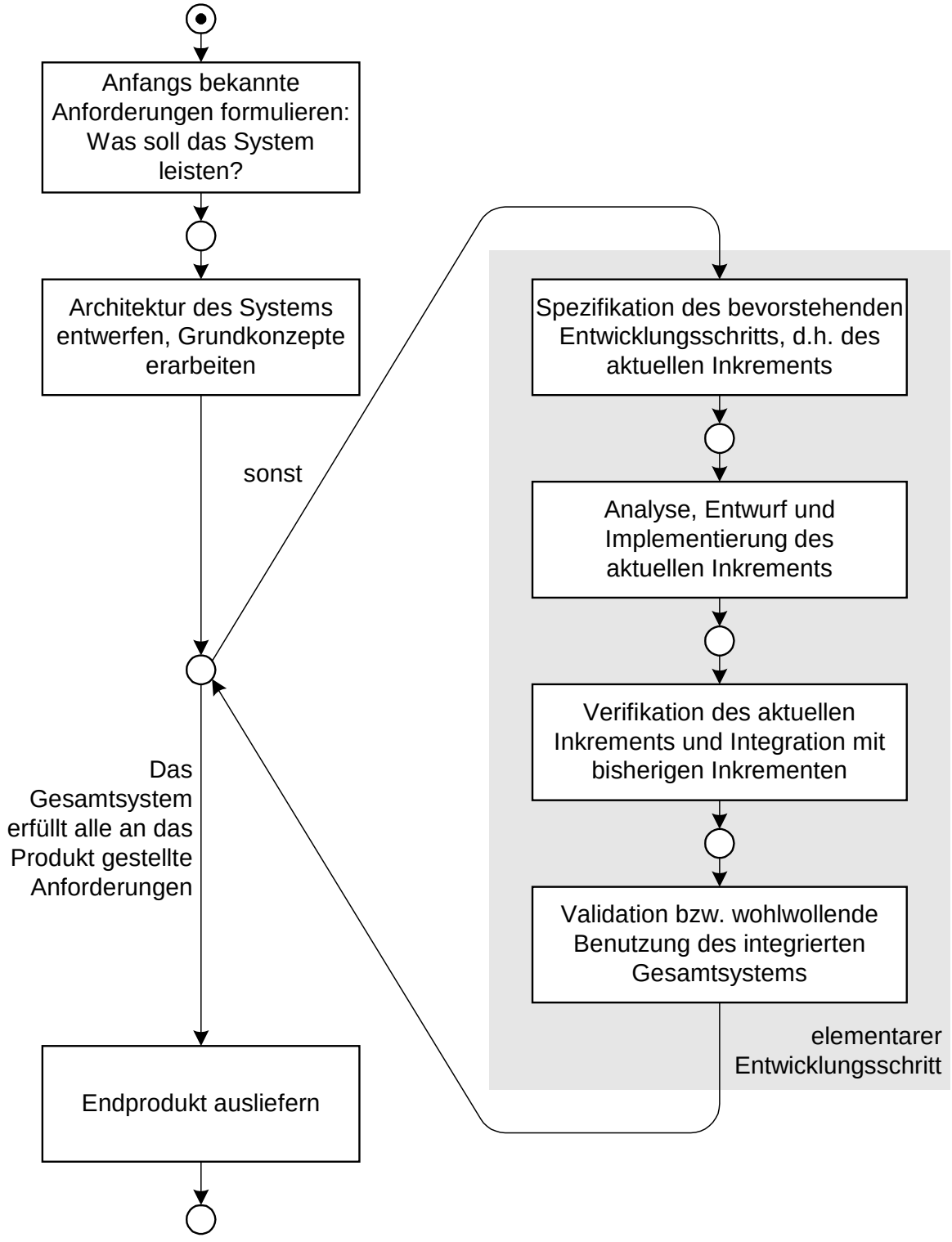

Abbildung 2–1: Der TimeLess–Entwicklungsprozeß

Die Qualität einer guten Architektur wird durch viele Faktoren beeinflußt Ein wichtiges Ziel beim Architekturentwurf ist das Erreichen eines hohen Grades an Unabhängigkeit – z.B. von Technologien. Dadurch bleibt die Austauschbarkeit und Wiederverwendbarkeit von Komponenten eher gewährleistet. Ein weiteres Qualitätsmerkmal guter Architektur wird durch eine Menge langlebiger Schnittstellen geprägt, die eine potentielle Systemmodifikation bzw.– veränderung ausschließlich durch Austausch oder Hinzunahme von Komponenten ermöglicht. Hierbei ist es nicht erforderlich die konkrete Ausprägung der Schnittstellen zu formulieren (Dies ist in dieser Phase meistens gar nicht möglich). Vielmehr gilt es deren Notwendigkeit zu erkennen und eine Abstraktionsebene[1] zu finden, auf der Eigenschaften so spezifiziert werden können, daß sie für alle Entwicklungsinkremente Gültigkeit behalten.

Sobald ein gewisses Maß an Zuversicht bezüglich der Eignung der Architektur– und Schnittstellenkonzepte innerhalb der Entwicklungsmannschaft vorhanden ist, sollte mit den iterativen Entwicklungsschritten begonnen werden. Hier ist sehr viel Fingerspitzengefühl notwendig, um sich nicht zu lange mit Konzeptarbeit zu beschäftigen, aber dennoch eine möglichst hochwertige Beschreibung der Architekturkonzepte zu erzielen. Dies hat für die später unerläßliche zwischenmenschliche Kommunikation innerhalb der Entwicklergruppe einen sehr hohen Stellenwert und ist außerdem für die Partitionierbarkeit der Gesamtanforderungen in Teilanforderungen der elementaren Entwicklungsschritte unabdingbar.

Die eingangs entworfenen Architekturkonzepte müssen außerdem noch nicht vollständig formuliert sein, da auch während der eigentlichen Entwicklungsphasen noch ein Rückfluß neuer bzw. ergänzender Erkenntnisse in den Architekturentwurf eingehen. Damit ist es möglich die Architektur eines Gesamtsystems stetig während des gesamten Entwicklungsverlaufs zu verfeinern und an zu Beginn noch unbekannte Begebenheiten anzupassen. Die Betonung liegt hierbei auf Verfeinerung, da man nach Möglichkeit bereits getroffene Entwurfsentscheidungen nicht (bzw. nur wenn Fehler entdeckt wurden) ändern sollte, da die Konsequenzen solcher Änderungen auf bereits vorhandene Entwicklungsinkremente verheerende Folgen haben können und somit die Qualität des Systems entscheidend verringern.

Die Vorgehensweise bei der Partitionierung des Gesamtprojekts in kleinere Entwicklungsschritte hängt sehr stark vom Anwendungsgebiet des Projekts ab. In größeren Projekten müssen meist erst trägersystemspezifische Dienste, die sich stark an den eingesetzten Technologien orientieren, implementiert werden, damit eine Integration von Entwicklungsinkrementen überhaupt möglich wird. Dieser Prozeß läßt sich jedoch normalerweise problemlos durchführen, da immer nur die Anforderungen der nächsten Etappe formuliert werden. Hierin liegt ein weiterer Vorteil dieses Entwicklungsprozeßmodells, denn eine konkrete Formulierung aller Anforderungen ist nicht unbedingt erforderlich. Man benötigt lediglich diejenigen, die für das aktuelle Inkrement relevant sind. Somit ist eine Anpassung an dynamische Gegebenheiten viel einfacher und flexibler möglich. Weiterhin kann dadurch die Entwicklungszeit reduziert werden, da das Anforderungsdokument des nächsten Inkrements bereits vor Fertigstellung des aktuellen Teilschritts ausgearbeitet werden kann (Nebenläufigkeit innerhalb des Entwicklungsprozesses). Dies wurde zur Vereinfachung in Abbildung 2–1 nicht gezeigt.

[1] vgl. [Bungert97]

Der eigentliche elementare Entwicklungsschritt erfolgt klassisch[1]:

1. Anforderungen an das Inkrement spezifizieren
2. Analyse und Entwurf des aktuellen Inkrements ⇒ Implementierungsspezifikation
3. Implementierung und Verifikation der in diesem Inkrement relevanten Komponenten
4. Integration der relevanten Komponenten in das bisherige Gesamtsystem.

Den letzten Schritt innerhalb einer Iteration stellt die Testphase dar. Dieser Schritt ist besonders wertvoll, da hier im Rahmen einer wohlwollenden Nutzung in Zusammenarbeit mit dem Auftraggeber (Kunde) geprüft werden kann, ob die bisherige Umsetzung des Systems auch tatsächlich die ursprünglich geforderten Eigenschaften besitzt. Hier kann also bereits sehr früh Fehlentwicklungen vorgebeugt werden. Darüber hinaus erhöht sich die Planbarkeit des weiteren Projektverlaufs entscheidend. Insbesondere sind die Entwicklungszeiten präziser ab– bzw. einschätzbar.

[1] In der Literatur wird dieser klassische Entwicklungsprozeß oft als Wasserfallmodell bezeichnet. Dieser wurde erstmals in [Royce70] näher charakterisiert.

Kapitel 3 Anforderungen

Die Entwicklung von Softwaresystemen im Umfeld großer Unternehmen stellt sicherlich eine der größten Herausforderungen in der Softwareentwicklung dar. Die Komplexität solcher Systeme erfordert oft sehr lange und teure Entwicklungs– und Einführungszyklen. Diese lassen sich nur durch einen langen und intensiven Produktiveinsatz der Systeme sowie einer dadurch bedingten Effizienzsteigerung im Unternehmen rechtfertigen lassen. Für Unternehmen ist die Langlebigkeit solcher Systeme eines der wichtigsten Kriterien. Doch was zeichnet ein langlebiges System aus? Die Umsetzung von Unternehmensanforderungen sowie die Möglichkeit, das System möglichst einfach und schnell an sich ändernde Gegebenheiten anpassen zu können, gehört zu den wichtigsten Eigenschaften solcher Systeme.

In diesem Kapitel werden zwei Klassen von Anforderungen unterschieden:

- Anwendungssemantische Anforderungen (TimeLess)
- Anforderungen an verteilte Mehr–Benutzer–Systeme

Anwendungssemantische Prozesse müssen sehr genau verstanden werden. Ebenso ist die Umgebung, in der das zu entwickelnde System eingesetzt wird, von großer Bedeutung, da sich nur dadurch allgemeine Anforderungen ableiten und verstehen lassen. Bei der Formulierung allgemeiner Anforderungen wurde immer wieder das TimeLess–Szenario verlassen, um eine Übertragbarkeit auf andere Anwendungsbereiche zu gewährleisten.

Bevor aus der TimeLess–Idee ein hochwertiges Produkt entwickelt werden kann, sollte man sich im klaren darüber sein, welche Vorgehensweise am zweckmäßigsten ist, um den besten Nutzen für das Projekt zu erzielen. Daher sollen die an diesen Entwicklungsprozeß gestellten Anforderungen als erstes behandelt werden.

3.1 Anwendungssemantische Anforderungen (TimeLess)

Ein klares Verständnis der Umgebung, in der ein zu entwickelndes Softwaresystem seinen Dienst verrichten soll, ist für dessen Erfolg unabdingbar. Viele aus unterschiedlichsten Blickwinkeln zu betrachtende Aspekte müssen bereits zu Beginn analysiert werden. Nur dadurch ist eine detaillierte und umfassende Vorstellung der Anforderungen an das Systems zu erlangen. Mehrere Anwendungsszenarien aus der Sicht unterschiedlicher Benutzerklassen müssen betrachtet werden, um ein hochwertiges Anforderungsbeschreibung anfertigen zu können. Allein diese Aufgabe stellt einen enormen Aufwand dar und wurde daher in den beiden getrennt abgefaßten Diplomarbeiten [Brand97] sowie [Langhauser97] ausführlich behandelt. Viele der in dieser Arbeit benutzten Begriffe setzen zum besseren Verständnis die Kenntnis der beiden Vorgängerarbeiten voraus.

Ziel dieses Abschnitts ist es, die bisher formulierten Wünsche an TimeLess zu konkretisieren um daraus die für das Abteilungsprodukt relevanten Anforderungen auszuarbeiten. Diese Anforderungen können in drei Gruppen aufgeteilt werden:

- Anforderungen des TimeLess–Kunden

  Der Kunde – beispielsweise ein Software herstellendes Unternehmen – verspricht sich durch den Einsatz von TimeLess eine Steigerung der Wirtschaftlichkeit seines Unternehmens. Dies kann sich in höherer Produktqualität, verbessertem Informationsfluß, Steigerung von Produktivität usw. äußern. Allerdings wird der Kunde auch einige nicht unbedingt absehbare, zum Teil sogar politische Rahmenbedingungen festlegen, deren Beachtung bei der Formulierung von Anforderungen an TimeLess unerläßlich ist.

- Anforderungen des TimeLess–Benutzers

  Der Benutzer ist derjenige, der während seiner täglichen Arbeit TimeLess „erlebt“. Seine Vorstellungen und Wünsche sollen insbesondere die Entwicklung der Wahrnehmungskomponenten maßgebend beeinflussen. Allerdings müssen auch die Einschränkungen des Abteilungsprodukts gegenüber dem Vollprodukt aufgeführt werden. Diese sind notwendig, um den zeitlichen und finanziellen Entwicklungsrahmen des Projekts nicht zu sprengen. Es wird versucht, diese Anforderungen nur an den Stellen einzuschränken, wo keine Gefahr einer falschen Entwurfsentscheidung befürchtet werden muß. Dadurch wird gewährleistet, daß funktionale Ergänzungen auch nachträglich im Rahmen des Entwurfskonzepts möglich sind.

- Anforderungen des TimeLess–Entwicklers

  Die Entwicklung von Softwaresystemen ist nur möglich durch den Einsatz geeigneter, dem Stand der Technik entsprechender Technologien. In ihren vielen Ausprägungen beeinflussen diese in hohem Maße den Erfolg bzw. Mißerfolg einer realisierten Sofwarelösung. Die Auswahl geeigneter Technologien muß sehr sorgfältig erfolgen. Dies setzt sehr gute Kenntnisse des technischen Umfelds voraus, in dem das zu entwickelnde Produkt eingesetzt werden soll. Insofern macht es zum Beispiel für die Entwicklung von TimeLess einen gewaltigen Unterschied, ob es sich um ein Ein-

benutzer– oder ein Mehrbenutzersystem handelt. Ebenso spielt es eine Rolle, ob TimeLess offen für den Anschluß fremder System bleiben muß (und damit Schnittstellen nach außen anbieten muß). Ein anderer Faktor ist die Verteilbarkeit von Systemkomponenten. Werden alle Komponenten nur auf einem Rechner oder auf vielen verteilt eingesetzt werden? Diese und andere Fragen gilt es vor der Entwicklung zu klären.

3.1.1 Anforderungen des TimeLess–Kunden

Unternehmen, deren Kapital zu großen Teilen aus dem Wissen der Mitarbeiter besteht, benötigen eine optimale Wissensverwaltung, bzw. ein optimales Wissensmanagement. Dies zeigen nicht nur mehrere Gutachten[1], sondern auch die gemachten Erfahrungen des Software Engineering Labors. Insbesondere Unternehmen im Bereich der Softwareentwicklung hängen immer mehr vom Wissen ihrer Mitarbeiter ab. Der Begriff „Corporate Memory“ bedeutet übersetzt „Unternehmensgedächtnis“. Ein solches Gedächtnis besteht in der Praxis allerdings noch allzu oft aus dem in den Köpfen der Mitarbeiter verankerten Wissen. Dies führt unter anderem zu folgenden Problemen, welche bei der Erstellung von Anforderungen an TimeLess zu berücksichtigen sind:

- Die Mitarbeiterleistung kann nur schwer eingeschätzt werden.

  Lösungsansatz: Das Abliefern von Wissen muß standardisiert werden. Nur abgeliefertes (in diesem Fall an TimeLess abgeliefertes) Wissen wird als Grundlage für eine Leistungsbewertung benutzt. Dies sollte jedoch nicht mit Kontrolle verwechselt werden. An TimeLess übergebene Dokumente sind Endprodukte, deren Entstehungsprozeß vom System nicht nachvollzogen werden kann und somit eine direkte Kontrolle nicht zuläßt.

- Bestimmte Aufgaben können nur von bestimmten Mitarbeitern erledigt werden.

  Wenn das Wissen in den Köpfen der Mitarbeiter verbleibt, entsteht eine in der Regel unerwünschte Abhängigkeit des Unternehmens von seinen Mitarbeitern. Diese können in Extremfällen das Unternehmen beispielsweise durch Kündigung oder überhöhte Forderungen in seiner Existenz gefährden.

  Lösungsansatz: Die Weitergabe des Wissens eines Mitarbeiters an einen anderen Mitarbeiter muß möglichst effizient geschehen und nach Möglichkeit auch ohne Dasein des Wissensträgers möglich sein. Dies kann dadurch realisiert werden, daß man Mitarbeitern ein Informationsverwaltungssystem zur Verfügung stellt, das sowohl durch die Natürlichkeit der Anwendung besticht, als auch durch die Freiheit das Wissen in einer nicht zwingend vorgeschriebenen Art und Weise ablegen zu müssen. Die Mitarbeiter sollen nicht zum Gebrauch von TimeLess gezwungen werden, sondern vielmehr aus eigenen Beweggründen heraus das Gefühl haben, zusammen mit TimeLess ihre Arbeit besser verrichten zu können. Dies führt sowohl für den Mitarbeiter

[1] Die im Rahmen von TimeLess in Auftrag gegebenen Gutachten [FZI97] und [IAO97] kommen zum Schluß, daß ein Informations– bzw. Dokumentenverwaltungssystem zur „verbesserten Versorgung der Mitarbeiter mit Wissen und Information“ hohe Relevanz besitzt.

als auch für das Unternehmen zu einer höheren Zufriedenheit und erlaubt es dem Unternehmen in viel höherem Maße von einem „Corporate Memory“ zu reden.

Kunden von TimeLess stellen darüber hinaus häufig noch Anforderungen, die nicht unmittelbar nachvollzogen werden können. Beispielsweise wird ein kleiner Kunde mit wenigen Mitarbeitern nicht einsehen, warum er eine TimeLess–Infrastruktur kaufen soll, die weit mehr leisten kann, als er benötigt. Dieses Problem ist sicherlich durch ein entsprechendes Lizenzierungskonzept lösbar, dennoch sollte man bei der Entwicklung von TimeLess berücksichtigen, daß es sowohl in sehr großen als auch in kleinen Unternehmen zum Einsatz kommen kann.

In diesem Zusammenhang muß auch die Unabhängigkeit von bestimmten Plattformen oder Trägersystemen genannt werden. Eine möglichst hohe Anpassungsfähigkeit an die jeweiligen Belange von Kunden und Benutzer ist eine entscheidende Eigenschaft bezüglich des Erfolgs von TimeLess. Anzustreben ist ein System, das auf allen gängigen Technologieplattformen lauffähig ist und dessen Nutzung nicht von einer bestimmten Rechnerumgebung abhängig ist. Für den Kunden bedeutet das, daß sich TimeLess in seine vorhandene EDV–Infrastruktur problemlos integrieren läßt. Erfordert TimeLess z.B. eine Datenbank, so sollte diese vom Kunden zur Verfügung gestellt werden können. Daher ist es auch wichtig, daß TimeLess nicht aus einem monolithischen Block besteht. Es ist vielmehr ein modulares Konzept anzustreben, welches ermöglicht, bestimmte Komponenten den speziellen Wünschen des Kunden anpassen zu können. Dadurch können bestimmte Komponenten gar vom Kunden selbst bereitgestellt werden.

Ein weiterer Aspekt, dessen Berücksichtigung sich TimeLess widmen muß, ist begründet durch die Mobilität moderner Unternehmen bzw. deren Mitarbeiter. TimeLess darf nicht davon ausgehen, daß es bestimmte vorkonfigurierte Arbeitsplätze gibt, die speziell für den Einsatz von TimeLess zur Verfügung gestellt werden. TimeLess, bzw. die Benutzerschnittstellenkomponente von TimeLess muß die Fähigkeit besitzen so mobil wie seine Benutzer zu sein, d.h. wenn ein Mitarbeiter auf einer Geschäftsreise ist, so will er auf die Funktionalität von TimeLess nicht verzichten müssen. Die Nutzung von TimeLess sollte daher nicht ortsgebunden sein. Insbesondere durch den hohen Verbreitungsgrad stellt das Internet für diese Anforderung sicherlich eine mögliche Technologiebasis dar, die eine darauf basierende Realisierung vereinfachen kann.

Die bisher erwähnten Aspekte beleuchten im wesentlichen die statische Struktur des Systems zum Zeitpunkt der Installation bzw. Inbetriebnahme. Ebenso ist es wichtig, Konzepte zu entwerfen, die eine Änderung bestimmter Strukturen zur Laufzeit vorsehen. So besitzt TimeLess zu Beginn beispielsweise nur die Fähigkeit, bestimmte Attribute bestimmter Informationseinheiten (z.B. Dokumente) zu verwalten. Der Kunde stellt jedoch während des Betriebs fest, daß Informationseinheiten in seinem Unternehmen mehr Eigenschaften besitzen, als es TimeLess vorsieht. Es muß also möglich sein, ein bestehendes TimeLess–Basissystem um diese neuen Eigenschaften zur Laufzeit erweitern zu können, ohne einen Neubau (neue Version) von TimeLess erforderlich zu machen. Daher muß eine Systemarchitektur gefunden werden, die die Änderbarkeit der Anwendungssemantik zuläßt und sie als Regel, nicht als Ausnahme betrachtet.

Weiterhin soll TimeLess auch offen bleiben für Erweiterungen durch fremde Systeme, wie z.B. die Integration eines Workflow–Managements[1]. Diese sind für die Durchführung längerer Vorgänge zuständig. Diese zeichnen sich darüberhinaus aus, daß sie Handlungen mehrerer Benutzer koordinieren. Beispielsweise sollen in TimeLess unter normalen Umständen keine Dokumente gelöscht werden können. Da dies in manchen Fällen trotzdem notwendig sein wird, muß TimeLess Verfahren unterstützen, die ein Löschen dennoch ermöglichen. Ein Beispiel hierfür ist, daß ein Dokument persönliche Daten enthält und diese aus rechtlichen Gründen nicht mehr gespeichert werden dürfen. Das Dokument muß folglich gelöscht werden. In diesem Fall könnte man sich z.B. vorstellen, daß das Dokument dann gelöscht werden darf, wenn mehrere Benutzer dazu einwilligen (beispielsweise ein Sachbearbeiter und dessen Vorgesetzter). Die allgemeine Spezifikation eines solchen Vorgangs, sowie die konkrete Durchführung im Bedarfsfall könnte durch ein in TimeLess integriertes Workflow–Management–System realisiert sein. Für den Benutzer ist wichtig, daß die natürliche Bedienbarkeit von TimeLess erhalten bleibt. Im Beispiel könnte man sich vorstellen, daß zum endgültigen Löschen eines Dokuments ein Löschanforderungsantrag ausgefüllt werden muß. Dieser muß von den Verantwortlichen unterzeichnet werden, bevor das Elemente gelöscht werden darf. Zusätzlich informiert er jedem Zeitpunkt alle Beteiligten über den Stand des Vorgangs.

3.1.2 Anforderungen des TimeLess–Benutzers

Die Hauptaufgabe von TimeLess aus der Sicht des Benutzers ist sicherlich der möglichst effiziente, intuitive Zugriff auf Informationen des Unternehmens. Effizient heißt in diesem Fall, daß der Benutzer unmittelbar an die gesuchte Information gelangt. Vielfach zwingen Informationsverwaltungssysteme Benutzer zu Handlungen, die technisch bedingt womöglich notwendig sind, aber dennoch dem Benutzer verborgen bleiben sollten. So wird der Benutzer meist gezwungen mit Begriffen wie „Festplatte“, „Betriebsystem“ oder „Netzwerklaufwerk“ umgehen zu können.

Intuitiv heißt, daß die dazu notwendigen Schritte von jedem Benutzer (auch dem Einsteiger) nachvollzogen werden können und möglichst selbst ohne langwierigen und kostenintensiven Schulungsmaßnahmen erlernt werden können. Einzige Voraussetzung für die Nutzung von TimeLess sollte der Umgang mit der allgemein bekannten Mensch–Maschine Schnittstelle (Tastatur, Maus o.ä.) sein.

Um dieses Ziel zu erreichen, soll durch TimeLess keine eigene, neu zu erlernende Welt geschaffen werden. Vielmehr soll eine bekannte Welt nachgebildet werden, die jeder Informationssuchende sicherlich schon kennt: die Bibliothek. Herkömmliche elektronische datenverwaltende Systeme haben oft die wenig anschauliche Eigenschaft, Informationen so zu präsentieren, wie sie auch technisch verwaltet werden. Beispielsweise redet man eher von Dateien und Verzeichnissen, statt von Büchern, Ordnern, Videofilmen oder anderen real existierenden Gegenständen. Ein HTML–Dokument besteht beispielsweise aus einer Textdatei, die wiederum Verweise auf eingebettete Grafiken enthält. Der Browser stellt diese Menge von Dateien jedoch als ein Dokument dar. Versucht der Benutzer nun dieses Dokument abzulegen (EDV–

[1] Workflow–Management (engl.): Spezifikation und Ausführen von Geschäftsprozessen, also betriebliche Abläufe innerhalb von Unternehmen, die insbesondere durch die Abhängigkeit von menschlichen Entscheidungsvorgängen gekennzeichnet sind.

Terminologie: zu speichern), so muß ihm klar sein, daß es sich hierbei um eine Sammlung mehrerer Dateien handelt. Das Speichern besteht somit aus mehreren, für den Benutzer schwer nachvollziehbaren Arbeitsschritten. Dadurch wird er gezwungen sich mit Vorgängen zu beschäftigen, die nichts mit seiner Zielsetzung zu tun haben.

TimeLess versucht dies zu verhindern, indem es Benutzern eine räumliche Darstellung dessen anbietet, womit bzw. worin sie gerade arbeiten. Den Benutzern soll eine vertraute Welt präsentiert werden, in der man sich sofort zurechtfinden kann. Man wird nicht mit Laufwerken, Anwendungsservern oder Netzwerken konfrontiert, sondern mit Bibliotheksgebäuden, die auf einem Campus in einem Land stehen. In diesen Gebäuden findet man Etagen, in denen Räume stehen, die voller Regale stehen können. Ähnlich einem Bücherregal, sind diese mit verschiedensten Dokumenten (z.B. Bücher, Videos, Prospekte) bzw. Sammlungsbehälter (wie z.B. Stehordner oder Sammelmappen) gefüllt. Die Anordnung und Gestaltung der Elemente dieser virtuellen Bibliothekswelt sollen möglichst individuellen Charakter besitzen, um die Wiederfindung eines dem Benutzer bekannten Ortes zu erleichtern. Ebenso gelangt der Benutzer so „beiläufig“, bzw. vorbeilaufend, in den Besitz zusätzlicher Information, die spätere Suchen erleichtern kann und sein natürliches Orientierungsgefühl allgemein verbessert.

Praktisch gelingt das nur, wenn dem Benutzer zu jedem Zeitpunkt bekannt ist, wo er sich befindet[1]. TimeLess soll Benutzern implizit ihren Standort in Form einer individuell gestalteten und für sie eindeutig identifizierbaren Umgebung mitteilen. Der Benutzer wird dadurch in die Lage versetzt, seinen Standort „beiläufig“ zu ermitteln, daß sich der Ort an dem er sich befindet von anderen Orten unterscheidet. Dies soll nicht nur durch die Bezeichnung des Ortes erfolgen, sondern auch durch die grafische Darstellung. So besitzen Strukturierungselemente (wie Regale und Dokumente) Attribute, die eine grafische Unterscheidung von anderen Elementen gleichen Typs zulassen. Dies kann z.B. die Farbe oder die Größe sein.

Benutzer des TimeLess–Informationslandschaft können in verschiedenen Benutzerrollen auftreten. Diese sind der Leser, der Bibliothekar und der Autor. Die Rolle bestimmt, wie ein Benutzer mit TimeLess interagiert. Um möglichst flexibel Benutzern Rollen zuordnen zu können, die ihnen in unterschiedlichen TimeLess–Bereichen eine bestimmte Nutzung ermöglichen, muß TimeLess ein Berechtigungswesen besitzen. Nachfolgend werden die Benutzerklassen näher charakterisiert:

- Der Leser

  In Anlehnung an die Begriffswelt einer Bibliothek, werden Benutzer in der Rolle des Lesers diejenigen sein, die das Informationsangebot nutzen. So wie man sich die Nutzung einer realen Bibliothek vorstellt, so kann auch der Leser durch die TimeLess–Welt gehen (navigieren) und Information in Form von multimedialen Dokumenten nutzen (lesen). Er ist nicht in der Lage aufgebaute Strukturen zu verändern, bzw. selbst Dokumente zu erstellen (publizieren). Es werden jedoch Möglichkeiten zur Verfügung gestellt, die dem Leser eine aktive Teilnahme am Aufbau und am Inhalt der Bibliothek ermöglichen. So kann er beispielsweise Kommentare überall da anbringen, wo er Änderungen für sinnvoll hält und damit sowohl konstruktive als auch destruktive Kritik üben. Somit ist eine Rückkopplung (Feedback) gegeben.

[1] Diese Eigenschaft ist z.B. bei dem derzeit hochaktuellen Medium für Informationssuche, dem Internet, nicht gegeben. Daher spricht man oft vom „Internet–Surfing“.

Kommentare werden für andere Benutzer sichtbar. So können z.B. Strukturierungsmängel dem zuständigen Bibliothekaren gemeldet, oder Fehler in einem technischen Dokument dem Autor des Dokuments mitgeteilt werden.

Ähnlich dem Ausleihmechanismus einer Bibliothek, kann ein Leser persönliche Kopien ganzer Dokumente mitnehmen. Darüber hinaus werden dem Leser weitere klassische Bibliotheksdienste, wie z.B. ein Suchregister, zur Verfügung gestellt.

- Der Bibliothekar

  In einer Bibliothek ist die Wiederauffindbarkeit eines Buches stark davon abhängig, ob deren Einsortierung einer für den entsprechenden Bereich gültigen Systematik unterworfen ist. Um dies garantieren zu können, werden Fachkräfte (Bibliothekare) speziell in diese Aufgabe eingewiesen. Diese sind auch die einzigen, die das Recht besitzen, neue Bücher einzusortieren bzw. vorhandene umzusortieren. In TimeLess können nur die Benutzer in einem bestimmten Bibliotheksbereich strukturelle Änderungen vornehmen, die das Bibliothekarsrecht für diesen Bereich besitzen. Damit wird vermieden, daß undurchsichtige Ablagestrukturen bedingt durch unterschiedliche Strukturierungskonzepte der verschiedenen Benutzer entstehen. Trotzdem bleibt die Freiheit, individuelle standardisierte Ablagestrukturen aufzubauen, für deren Umsetzung eine geschulte Gruppe von Bibliothekaren verantwortlich ist. TimeLess soll somit nicht das Strukturierungskonzept vorgeben, sondern lediglich Werkzeug zur Gestaltung und Pflege von individuell festlegbaren Bibliotheksstrukturen sein.

- Der Autor

  Der Bedarf eines Informationsverwaltungssystems entsteht durch die gewaltige Anhäufung von Information unterschiedlichster Art innerhalb eines Unternehmens. Diese Information liegt dabei immer mehr als multimediales Dokumente vor, wie Bild– oder Tonbandaufzeichnungen. Die Verwaltung von Dokumenten als Träger jeglicher Information gehört zu den Aufgaben von TimeLess. Dokumente werden jedoch nicht in TimeLess erstellt. Vielmehr werden sie durch Autoren mit Hilfe TimeLess–fremder Werkzeuge außerhalb erzeugt. Ähnlich einer realen Bibliothek, müssen Dokumente von außen an TimeLess übergeben werden, bevor diese mittels TimeLess–Techniken verwaltet werden können. Derjenige, der Dokumente erstellt, kann sie in der Rolle des Autors an TimeLess übergeben. Dabei kann er sie mit einem bestimmten Einsortierungsvorschlag versehen, der die Ablage für den Bibliothekaren erleichtern kann. Die Verantwortung für die Ablage von Dokumenten verbleibt jedoch bei dem/den zuständigen Bibliothekaren. Der Autor eines Dokuments kann in TimeLess jederzeit ermittelt werden, damit beispielsweise Kommentare an ihn weitergeleitet werden können.

Die bisher angesprochenen Anforderungen und geforderten Eigenschaften von TimeLess deuten sehr stark auf eine am Bildschirm wahrnehmbare, virtuelle Multimediabibliothek hin. Dies ist der Natürlichkeit wegen auch erwünscht. Allerdings werden an TimeLess auch darüber hinausgehende Forderungen gestellt, die insbesondere die täglichen Routinearbeiten er-

leichtern sollen. So besitzt z.B. jeder Benutzer von TimeLess sein eigenes Büro, in dem er und nur er alle bisher dargestellte Rollen übernehmen kann. Im Büro kann ein Benutzer Dokumente erstellen (Autor), er kann sie einsortieren, umsortieren und löschen (Bibliothekar) sowie nutzen (Leser). Das Büro eines Benutzers gilt als absolute Privatsphäre, in der er tun und lassen kann was er will (sofern es vom System unterstützt wird). Dem TimeLess–Benutzer darf auf keinen Fall der Eindruck entstehen, daß er kontrolliert oder beobachtet wird. Nur so wird er genügend Vertrauen besitzen, um TimeLess langfristig einzusetzen.

3.1.3 Anforderungen des TimeLess–Entwicklers

TimeLess sollte nicht als akademisches Forschungssystem betrachtet werden. Das Ziel von TimeLess ist in der ersten Phase die Entwicklung eines Abteilungsprodukts, welches im „Inselbetrieb“ unter „wohlwollender“ Benutzung zum Einsatz kommen soll. Dabei sollen auf bestimmte, für ein späteres Produkt notwendige Merkmale nicht verzichtet werden. Ebenfalls ist es wichtig, praktisch verfügbare, hinreichend verbreitete Technologien einzusetzen, die eine pragmatische Vorgehensweise während der Entwicklung ermöglichen. Leider müssen beim derzeitigen Stand einsetzbarer Technologien, wie z.B. Softwareentwicklungsumgebungen, immer wieder Unzulänglichkeiten in Kauf genommen werden. Diese zwingen manchmal zu einer nicht hundertprozentig zufriedenstellenden Lösung. In diesen Fällen ist es für die Qualität des Produkts wichtig, die zugrundeliegenden Konzepte sauber zu dokumentieren, damit an der entsprechenden Stelle bei Verfügbarwerden der Technologie nachgebessert werden kann. Insofern soll die TimeLess–Entwicklung durch den Versuch geprägt sein, Konzepte unabhängig von Technologien zu erarbeiten und trotzdem eine bestmögliche Realisierung mit vorhandenen technischen Mitteln durchzuführen.

Bei der Umsetzung bzw. konkreten Implementierung der Konzepte sollte grundsätzlich überlegt werden, ob der Einsatz bereits existierender, am Markt zu erwerbenden Produkte oder Teilprodukte in Frage kommt. In Anbetracht des zu erwartenden hohen Entwicklungsaufwands sollten möglichst viele Komponenten von TimeLess gekauft werden, statt sie selbst neu zu entwickeln. Dies gilt insbesondere für die GUI–Controls[1] sowie für die Datenbankkomponente. Beim letzteren wurde konkret schon im Vorfeld die Nutzung der relationalen Datenbank Adabas–D der Firma Software AG vorgegeben, da diese für das Abteilungsprodukt kostenfrei eingesetzt werden kann und im Entwicklungsumfeld ausreichende Produktkompetenz besteht. Diese Gegebenheit sollte das Projekt jedoch nicht weitergehend einschränken, da der Einsatz jeder beliebigen SQL–Datenbank[2] mit genormter Schnittstelle zu den nicht–funktionalen Anforderungen von TimeLess gehört.

Weitere Anforderungen an die Entwicklung sind eher als allgemein anzusehen (nicht TimeLess–spezifisch) und werden somit in dem folgenden Abschnitt behandelt.

[1] GUI = (engl.) Graphical User Interface, d.h. grafische Benutzerschnittstelle. Hierbei wird meistens von Bildschirmelementen gesprochen, die eine Interaktion mit dem Benutzer mittels Maus und Tastatur erlauben. Beispiele dieser sog. GUI–Controls sind Knöpfe, Listenfelder, Menüs, Fenster (Windows) usw.

[2] SQL = (engl.) Structured Query Language, eine deskriptive Datenbank–Abfragesprache, die sich als weltweiter Standard etabliert hat. Fast jedes relationales Datenbanksystem besitzt eine SQL–Schnittstelle.

3.2 Anforderungen an verteilte Mehr–Benutzer–Systeme

Da es sich bei TimeLess um ein Softwaresystem handelt, das insbesondere auch für den Einsatz in großen Unternehmen konzipiert werden soll, ist es wichtig Merkmale und Eigenschaften zu nennen, die diese Klasse von Softwaresystemen kennzeichnen. Daran werden sich die wesentlichen architekturellen Konzepte von TimeLess zu orientieren haben. Großer Wert besitzt zudem die Wiederverwendbarkeit der zu erarbeitenden TimeLess–Architektur in anderen Softwaresystemen dieser Klasse.

3.2.1 Mehr–Benutzer–fähig

TimeLess hat primär den Zweck eine Vielzahl von Mitarbeitern eines Unternehmens bei der Informationsbeschaffung und –verwaltung zu unterstützen. Insofern ist es wichtig festzuhalten, daß es sich hierbei um ein Mehr–Benutzer–fähiges System handelt, d.h. daß mehrere Benutzer gleichzeitig mit dem System arbeiten können. So wie auch viele Bibliotheksbenutzer sich gleichzeitig in einer Bibliothek aufhalten und deren Informationsangebot nutzen können, so sollen auch TimeLess–Benutzer jederzeit über ihre Benutzerschnittstelle Dienste von TimeLess nutzen können. Abbildung 3–1 zeigt den allgemeinen Aufbau eines Mehr–Benutzer–fähigen Informationsverwaltungssystems, welches dem Benutzer durch den Systemverwalter Dienste über die Benutzerschnittstelle zur Verfügung stellt. Der Systemverwalter muß hierbei in der Lage sein, Aufträge von vielen Benutzern entgegenzunehmen und diese abzuarbeiten.

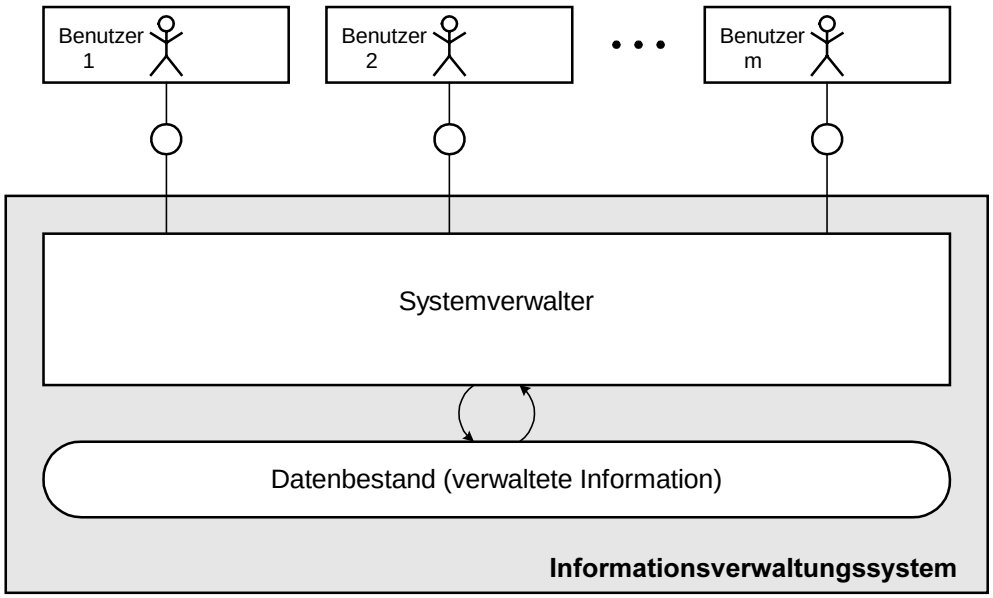

Abbildung 3–1: Das Mehr–Benutzer–fähige Informationsverwaltungssystem

Gewöhnliche Systeme gehen hier oft den Weg, daß sie versuchen jedem Benutzer den Eindurck zu vermitteln, er sei der einzige Anwender, der das System benutzt. Dies nennt man auch den logischen Einbenutzerbetrieb. Da jedoch jedes dieser „persönlichen Einbenutzersysteme“ letztlich auf gleiche Daten zugreift, ist es notwendig eventuell konkurrierende, also gleichzeitig stattfindende Zugriffe zu sequenzialisieren. Dies erreicht man in dem man die zu

verändernden Datenbereiche sperrt und erst nach erfolgreicher Überführung in den neuen Zustand wieder für weitere Nutzung anderer freigibt. Die hierfür in der Praxis eingesetzten Transaktionskonzepte[1] sind sowohl Benutzer als auch für Entwickler oft nicht unmittelbar nachvollziehbar. Ein Benutzer, dem nicht bewußt gemacht wird, daß das System auch gleichzeitig von anderen benutzt werden kann, hat kein Verständnis für nicht durchführbare Operationen bedingt durch Zugriffskonflikte. Lösungen dieser Problematik sollen im Rahmen von TimeLess durch neue Konzepte gesucht werden.

3.2.2 Verteilbarkeit

Die Anforderungen an moderne informationstechnische Systeme haben sich durch die zunehmende Vernetzung und Globalisierung von Unternehmen stark verändert. Der unternehmensweite Zugriff auf verteilte Datenbestände gehört zu den wichtigsten Eigenschaften heutiger Informationssysteme. Aber auch über Unternehmensgrenzen hinweg haben Entwicklungen, wie z.B. das Internet, zu weltweit vernetzten Informationssystemen geführt. Einige Beweggründe Informationssysteme als verteilte Systeme, d.h. als Verbund vieler unabhängiger Teilsysteme zu konzipieren, werden nachfolgend aufgelistet:

- Dezentrale Unternehmensstrukturen

  Dezentrale Unternehmensstrukturen erfordern häufig auch eine dezentrale Datenverwaltung. In einem Versandhaus z.B. werden die Personalakten an einer physisch anderen Stelle abgelegt als die Lagerbestandslisten. Ein Unternehmen, das TimeLess einsetzen möchte, könnte z.B. auch verlangen, daß für jedes seiner Zweigstellen ein Campus innerhalb des TimeLess–Lands eingerichtet wird, dessen Inhalt von den jeweiligen Zweigstellen gepflegt und physisch verwaltet wird. Wer würde schon einsehen, daß ein Dokument, das in einer deutschen Filiale erstellt wurde und auch primär dort genutzt wird, in einer weit entfernten zentralen Dokumentaufbewahrungsstelle abgelegt wird? Das wäre in etwa so, als wenn es in Deutschland nur eine Bibliothek gäbe, die alle Bücher Deutschlands aufbewahren soll. Jeder Ausleihvorgang wäre mit dem Gang zu dieser Zentralstelle verbunden. Dieses Szenario ist realitätsfremd und somit auch für Datenverwaltungssysteme nicht sinnvoll.

- Skalierbarkeit

  Moderne Unternehmen zeichnen sich durch eine hohe Dynamik aus – meist verbunden mit der Notwendigkeit sich am Markt zu orientieren. Dadurch bedingtes Wachstum erfordert ein Mitwachsen der eingesetzten Softwaresysteme. Allerdings stößt jedes System irgendwann an Grenzen. Ein Datenbanksystem, z.B. schafft nur eine bestimmte Anzahl von Transaktionen pro Sekunde. Eine Verbesserung des Problems kann ein Verteilungskonzept bringen. Statt die Leistungsfähigkeit einer bestimmten Systemkomponente zu erhöhen, wird die Anzahl der Komponenten erhöht. So wie ein Unternehmen neue Mitarbeiter einstellt, wenn die vorhandenen an ihre Belastungsgrenzen stoßen, so sollte auch ein Softwaresystem durch bloßes „hinzufügen“ von Komponenten eine bessere Anpassungsfähigkeit ermöglichen. Dies setzt

[1] Eine Transaktion ist eine ununterbrechbare Folge von Operationen, welche ein System (Bsp.: Datenbank) von einem logisch konsistenten Anfangs– in einen logisch konsistenten Endzustand überführt.

allerdings zusätzliche Fähigkeiten der Systemkomponenten voraus, die im Rahmen der Architekturüberlegungen behandelt werden müssen.

- Fehlertoleranz, redundante Systemkomponenten

  Viele Informationssysteme können nur dann ihrer Funktion entsprechend arbeiten, wenn alle Teile des Systems fehlerfrei laufen. Der Absturz von Teilsystemen führt häufig dazu, daß das komplette System nicht mehr lauffähig ist. Die Fehlertoleranz, bzw. das Einplanen von Fehlverhalten innerhalb eines Systems, ist in der Regel sehr gering. Man stelle sich als Vergleich vor, daß ein ganzes Unternehmen lahmgelegt wird, wenn ein einziger Mitarbeiter erkrankt – ein undenkbares Szenario. Statt dessen wird das Unternehmen eine Lösung für solche Fälle bereits konzipiert haben. Es könnte z.B. die Arbeit des kranken Mitarbeiters auf andere Mitarbeiter übertragen, oder es könnte für diese Zeit ein Ersatzmitarbeiter eingestellt werden. Alternativ dazu könnte die Arbeit auch bis zu dessen Genesung liegen bleiben.

  Diese Fähigkeiten, übertragen auf Softwaresysteme implizieren, daß bestimmte Komponenten eines Systems bei Ausfall ersetzt, bzw. deren Aufgaben von anderen Komponenten übernommen werden. Hierbei ist zu sagen, daß insbesondere auch die Abwicklerkomponenten mehrfach vorhanden sein müssen. Somit muß sowohl eine physische als auch eine logische Komponentenredundanz vorgesehen sein.

- Online–Upgrade

  Um eine Erweiterung der Systemfunktionalität durchführen zu können aber auch um Systemfehler bzw. –mängel zu beheben, ist es wichtig, daß Teile des Systems jederzeit ersetzt werden können. Dies muß für Benutzer möglichst transparent geschehen und während des laufenden Betriebs möglich sein. Dieser sogenannte Online–Upgrade stellt insbesondere in Kombination mit der bereits genannten Fehlertoleranz eine Anforderung dar, die insbesondere bei großen Systemen als allgemeine Forderung gilt. Auch in der Realität würde man die Produktion in einer Firma nicht unterbrechen wollen, nur weil eine zusätzliche Maschine aufgebaut und installiert wird. Daher ist es sinnvoll, Softwaresysteme als Verbund unabhängiger, jederzeit austauschbarer Teilsysteme zu sehen.

3.2.3 Natürlichkeit

Die Fähigkeit eines Systementwicklers die Komplexität seines Aufgabengebiets zu beherrschen wird bei der Entwicklung von Systemen häufig nicht ernsthaft in Frage gestellt. Auch in anderen Branchen hängt die Qualität eines Produkts in entscheidendem Maß davon ab, wie gut die an der Entwicklung des Produkts beteiligten Mitarbeiter ihr Arbeitsfeld beherrschen. Oft wird dies auch beeinflußt durch die Qualität der Aufgabenteilung, also wie gut große Aufgaben in kleinere Häppchen aufgeteilt und entsprechend delegiert werden. Dadurch kann die Komplexität der jeweiligen Teilaufgaben reduziert werden, damit sie für die Ausführenden beherrschbar wird. Es ist wichtig, dabei eine gemeinsame Sprache zu finden, mit der die Entwickler untereinander kommunizieren. Dadurch können Mißverständnisse leichter vermieden werden. Die Entwickler sollten mit dem Anwendungsfeld, aber auch mit den ver-

schiedenen Darstellungsarten bzw. Plantypen vertraut gemacht werden, die während der Entwicklung zum Einsatz kommen. Es ist wichtig, daß in jeder Phase der Entwicklung auch Bezeichnungen gefunden werden, die möglichst präzise die Semantik des Bezeichneten beschreiben. So werden für aktive Komponenten (Akteure) meistens Begriffe gesucht, die an menschlichen Tätigkeiten angelehnt sind, wie z.B. Verwalter, Manager oder Übersetzer. Passive Komponenten (Speicher) hingegen kann man oft sehr anschaulich als Formular oder Papier bezeichnen, das als Träger für bestimmte Daten benutzt werden kann. Hier spielt die Natürlichkeit der Bezeichnungen eine große Rolle, denn in der Regel sind Begriffe aus einem bekannten Umfeld (aus dem realen Leben) leichter nachvollziehbar.

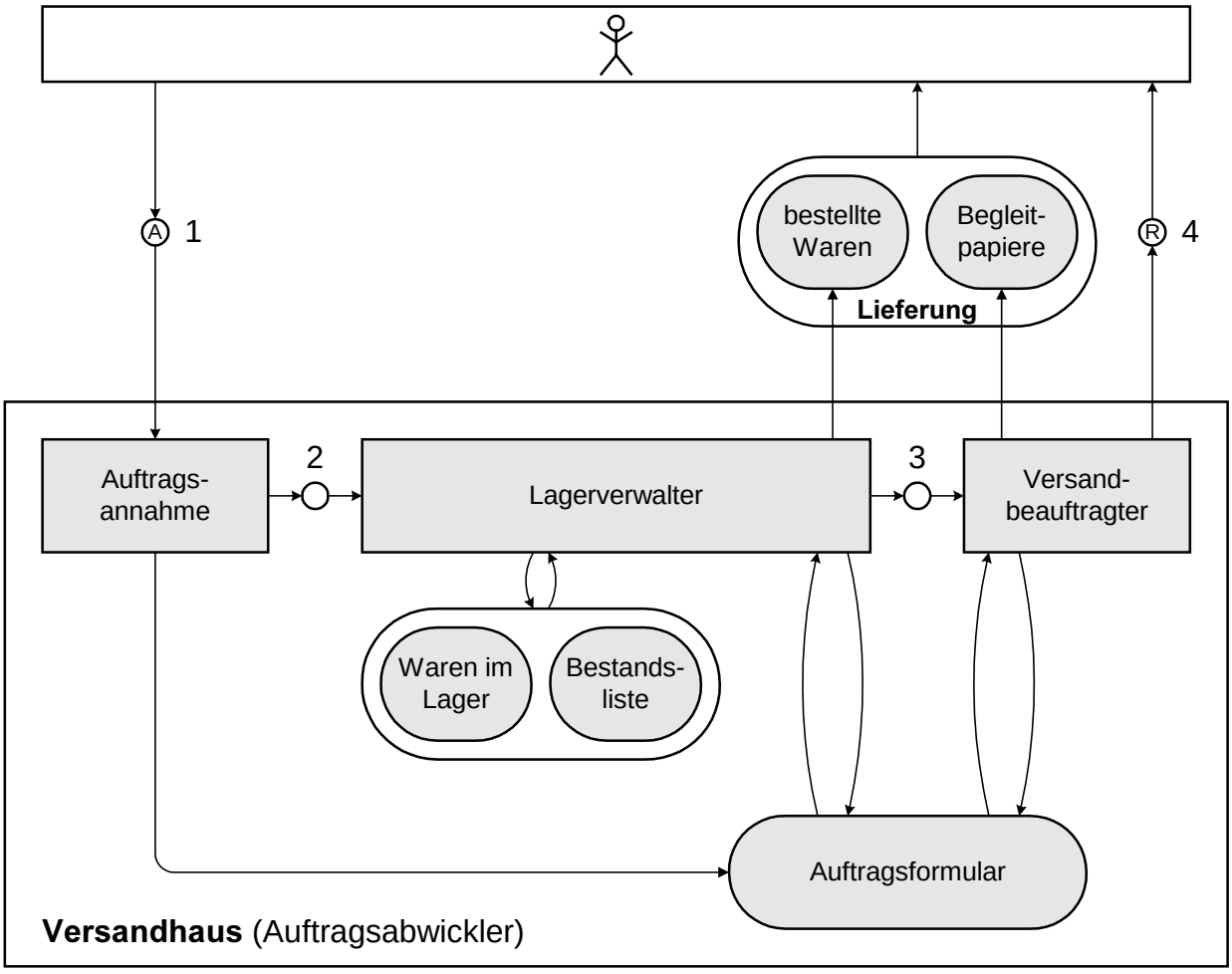

Abbildung 3–2: Das Versandhaus als Auftragsabwickler

Als Beispiel zeigt Abbildung 3–2 den Aufbau eines Versandhauses. Es besteht aus mehreren Mitarbeitern (Akteuren), die bestimmte Funktionen innerhalb des Unternehmens ausüben. So nimmt beispielsweise ein Mitarbeiter telefonisch Aufträge (z.B. Bestellung von Waren) eines Kunden entgegen, die in einem Auftragsformular (Speicher) eingetragen werden. Das Auftragsformular wird daraufhin an einen weiteren Mitarbeiter weitergereicht, der den Lagerbestand prüft. Dieser Lagerverwalter wirft einen Blick auf seine Lagerbestandsliste – ebenfalls ein Speicher. Die entsprechenden Warenmengen in dieser Liste vergleicht er mit der gewünschten Stückzahl des Auftrags. Bei ausreichendem Lagerbestand entnimmt er die entspre-

chenden Waren dem Lager. Abschließend erstellt der Versandbeauftragte die notwendigen Begleitpapiere wie Lieferschein und Rechnung und macht das Paket bereit zur Übergabe an den beauftragenden Kunden. Der gesamte Vorgang wird begleitet durch das ursprünglich bei der Auftragsannahme erstellte Auftragsformular, welches jedem beteiligten Mitarbeiter als Informationsquelle zur Verfügung steht. Dieses wird auch benutzt um den aktuellen Auftragszustand zu protokollieren.

Im Aufbaubild wurde die Reihenfolge, in der die jeweiligen Personen miteinander kommunizieren[1] entsprechend numeriert, um den soeben beschriebenen Vorgang der Auftragsabwicklung besser nachvollziehen zu können. Auf die Darstellung des Ablaufs in Form eines Petrinetzes wurde bedingt durch die reine Sequenz des Vorgangs verzichtet.

Das oben dargestellte Beispiel soll darstellen, daß technische Zusammenhänge oft durch natürliche Modelle vereinfacht werden und zum Verständnis beitragen können. Das Beispiel kann als Modell zur abstrakten Darstellung der Auftrags–/Rückmeldesemantik angesehen werden. Diese Semantik kommt bei der objektorientierten Modellierung beispielsweise sehr häufig vor. Im Bild ist dies erkenntlich gemacht durch Bezeichnung der beiden Kanäle vom und zum Kunden, die jeweils mit „A“ für Auftrag und „R“ für Rückmeldung gekennzeichnet sind. Dieser und auch andere Zusammenhänge lassen sich viel besser vermitteln, wenn man sie versucht durch natürliche Modelle darzustellen, die einen klaren Bezug zur Realität besitzen. Dies stellt ein wichtiger Aspekt bei der Entwicklung von TimeLess dar.

In Abbildung 3–2 erkennt man aber auch, daß die Aufteilung von großen komplexen Aufgaben in kleinere Zuständigkeitsbereiche ein sehr natürlicher Vorgang darstellt. Höchstens kleine unstrukturierte Firmen würden alle drei dargestellten Funktionen (Auftragsannahme, Lagerverwaltung, Versand) einem einzigen Mitarbeiter (einem Akteur) anvertrauen. Versetzt man sich in die Rolle des Geschäftsführers des Versandhauses, so findet man schnell Gründe, die eine solche Aufteilung in unterschiedliche Teilaufgaben sinnvoll erscheinen lassen. In Tabelle 1 sind in der ersten Spalte mehrere solche Gründe genannt. Die zweite Spalte stellt die entsprechenden Merkmale einer unter diesen Natürlichkeitsaspekten entworfenen Softwarekomponente dar. Diese Gegenüberstellung verdeutlicht die Ähnlichkeit von Softwarekomponenten und Gegebenheiten der Realität.

Diese Vorgehensweise bei der Konzeption, Spezifikation und Konstruktion von Softwaresystemen mag als unüblich erscheinen, sie dient jedoch dem Problemverständnis. Insbesondere wird Entwicklern in den Projekten geholfen, in denen eine sehr effektive Kommunikation zum Verständnis notwendig ist. Dazu zählen beispielsweise die Projekte, in denen eine formale Problembeschreibung (Anforderungsbeschreibung) nicht möglich ist. Dadurch erreicht man, daß auch komplexe Systeme mit einer hohen Qualität entwickelt werden und darüber hinaus beherrscht werden können.

[1] Die Kommunikation zwischen Akteuren wird durch Kanäle dargestellt. Diese unterstützen sowohl ereignis- als auch wertorientierte Kommunikation.

Versandhaus	Softwarekomponente
• Die Mitarbeiterqualifikation beschränkt sich auf das, was für die jeweilige Tätigkeit gebraucht wird.	• Die Komplexität von Akteuren wird beschränkt auf das Notwendige. Dadurch läßt sich deren Verhalten leichter beschreiben aber auch leichter verstehen.
• Jeder Aufgabenbereich kann vom jeweiligen Mitarbeiter optimiert werden, da er nur in seinem Bereich die Verantwortung trägt.	• Lokale algorithmische Optimierungen können bei Bedarf nachträglich gemacht werden, falls beispielsweise schlechte Performance zu unbefriedigenden Ergebnissen führt.
• Die Teilaufgaben können aus organisatorischen Gründen zu jedem Zeitpunkt ersetzt bzw. ergänzt werden. So wäre es denkbar, daß die Lagerverwaltung von einem fremden Unternehmen übernommen wird.	• Teilkomponenten können aus unterschiedlichsten Gründen ausgetauscht werden. So ist es denkbar, daß z.B. eine Komponente gegen eine mit gleichem Verhalten aus politischen Gründen ersetzt wird (z.B. weil der Kunde dies wünscht).
• Dem Kunden bleiben Interna des Versandhauses unbekannt. Die innere Struktur des Versandhauses kann jederzeit verändert werden, ohne dies dem Kunden mitteilen zu müssen. Wenn das Versandhaus aus nur einem Mitarbeiter bestehen würde, dann könnte das der Kunde nicht feststellen.	• Daten– und Strukturkapselung (Information hiding). Von außen wird über eine festgelegte Schnittstelle ein bestimmtes Verhalten erwartet. Welche Strukturen zur Umsetzung dieses Verhaltens notwendig sind, bleibt verborgen.
• Nur die Auftragsannahme steht in direktem Kontakt zum Kunden. Sie pflegt das Image des Versandhauses gegenüber dem Kunden (Repräsentant). Somit ist es wichtig, daß sie möglichst kundenorientiert handelt, d.h. eher dem Kunden etwas bietet, statt etwas von ihm zu verlangen.	• Das von einer Komponente über seine Schnittstelle angebotene Dienstrepertoire sollte vollständig sein. In vielen Fällen bietet es sich an, ein gewisses Maß an Redundanz vorzusehen. Dies ermöglicht Benennungen bzw. Signaturen, die viel besser die Semantik der Dienste ausdrücken können und somit sich fast von selbst dokumentieren.

Tabelle 1: Vergleich Versandhaus (Realität) mit gewünschten Softwareeigenschaften

3.2.4 Trennung von Zuständigkeiten ⇒ „Separation of Concerns“

Das soeben beschriebene Versandhaus zeigt bereits, wie natürlich die Aufteilung eines Gesamtaufgabenkomplexes in verschiedene Teilkomplexe sein kann. Diese Natürlichkeit entsteht dadurch, daß jedem Akteur klare Aufgaben bzw. Zuständigkeiten (engl.: concerns) zugeordnet werden können. Um beim architekturellen Entwurf von Softwaresystemen eine solche Zuordnung von Zuständigkeiten zu Komponenten oder Akteuren machen zu können, ist die Identifikation und Unterscheidung verschiedener „Concerns“ notwendig. Dieser Prozeß hängt allerdings stark von der Erfahrung des/der Systemarchitekten ab, da zu Beginn des Entwurfs oft nicht alle Zuständigkeitsbereiche explizit bekannt sind. Als Hilfsmittel können wieder die bereits existierenden Strukturen großer Unternehmen, wie z.B. die eines Versandhauses, zu Rate gezogen werden. Diese sind meist durch einen langen, evolutionären Prozeß gewachsen und enthalten dadurch sehr viele Erkenntnisse, die man sich bei der Identifikation von Zuständigkeiten innerhalb eines Softwaresystems zunutze machen kann.

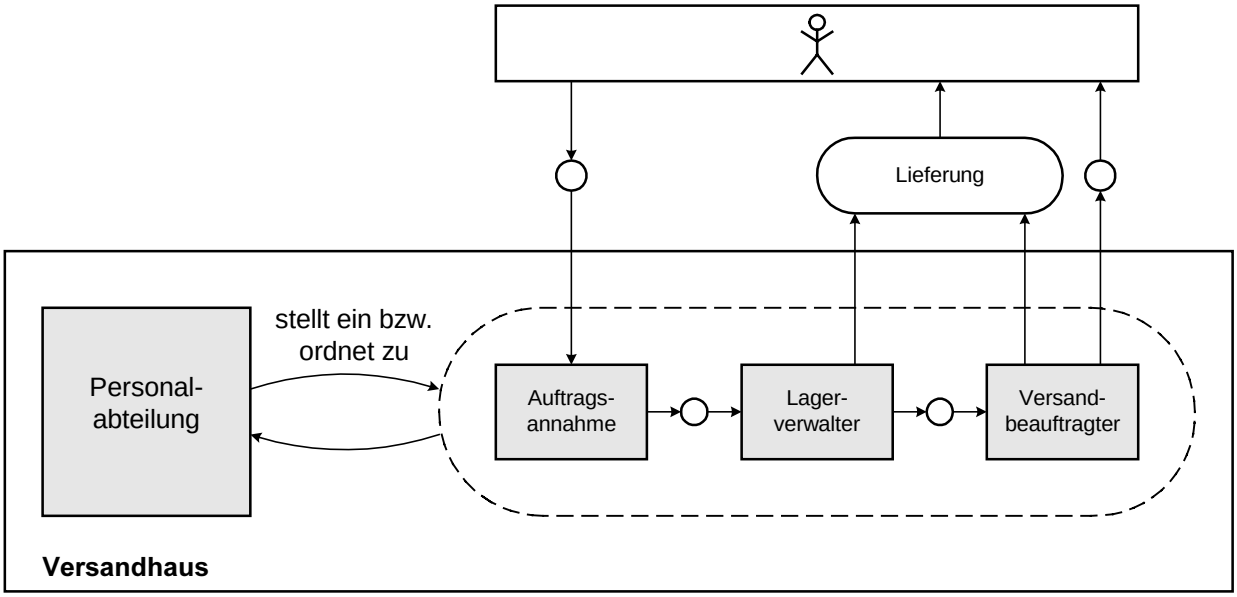

Abbildung 3–3: Strukturvarianz im Versandhaus

Als Beispiel soll an dieser Stelle eine Abteilung genannt werden, die einen speziellen Zuständigkeitsbereich beherbergt, der in fast jedem Unternehmen vorkommt: die Personalabteilung. Die Personalabteilung spielt in jedem größeren Unternehmen eine wichtige Rolle, obwohl sie normalerweise nicht direkt zum Geschäftsergebnis beiträgt. Sie ist vielmehr dafür zuständig, daß die Mitarbeiterstruktur stets optimal den betrieblichen Erfordernissen angepaßt ist. So müssen z.B. bei erhöhter Kundennachfrage weitere Mitarbeiter eingestellt, bzw. vorhandene besser oder in anderer Form eingesetzt werden. Dieser Zusammenhang wird in Abbildung 3–3 dargestellt. In diesem Bild sieht man, daß die Personalabteilung durch den Einstellungsvorgang die Strukturvarianzverwaltung im Unternehmen übernimmt. Das bedeutet, daß Strukturen (in diesem Fall Mitarbeiterstrukturen) geschaffen, geändert und gelöscht werden. Wie in großen Unternehmen, wird auch beim Entwurf komplexer Softwarearchitekturen der Aspekt der Strukturvarianz eine Rolle spielen.

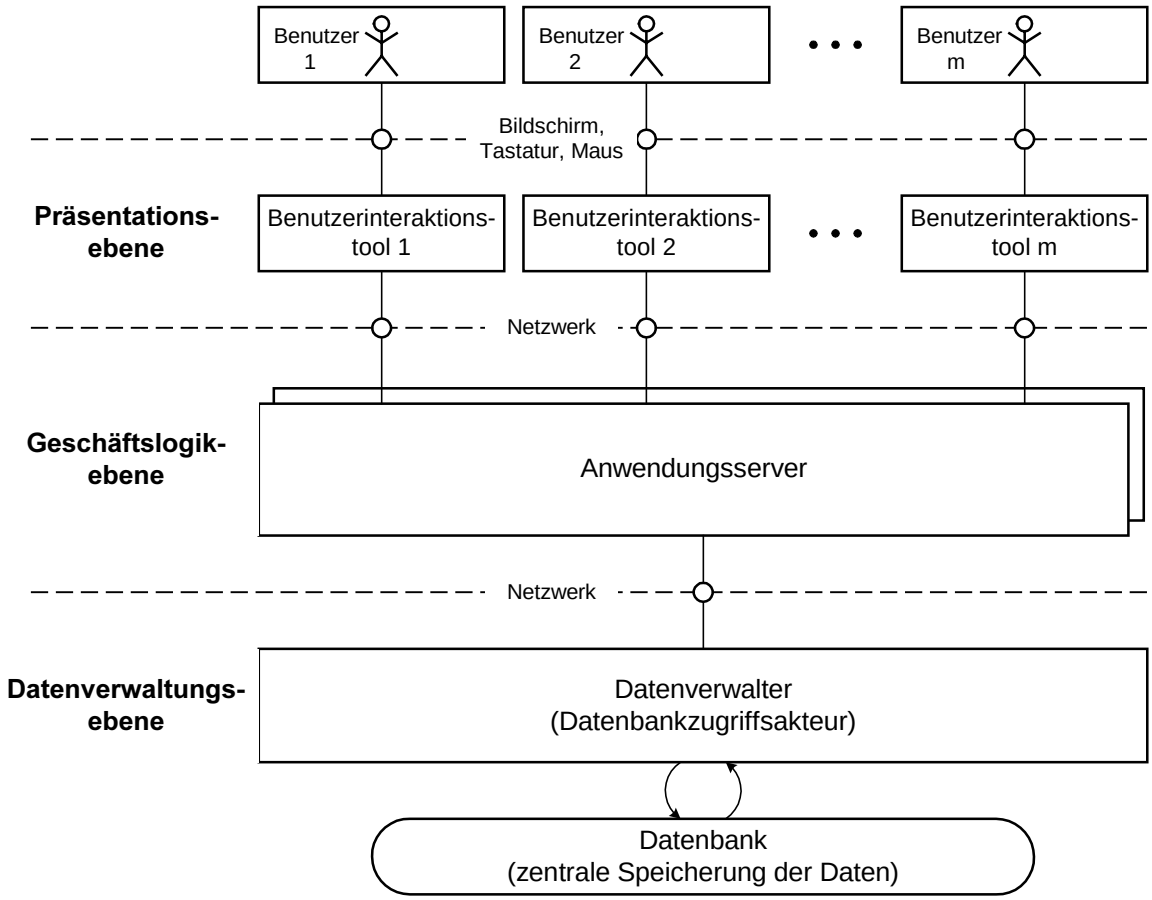

Abbildung 3–4: Die 3–Ebenen Architektur bei Client/Server Systemen

Drei weitere „Concerns“ kann man aus einem vorhandenen allgemeinen Architekturmodell ableiten, das sich als Grundgerüst für Systeme dieser Klasse bereits vielfach bewährt hat. Diese sogenannte 3–Ebenen Architektur (engl.: 3–tier architecture) trennt die Aspekte Präsentation, Geschäftslogik und Datenverwaltung. Diese Architektur entstand im Zuge der Entwicklung von Client/Server Systeme[1] und ordnet jedem Aspekt eine physikalische Ebene zu. Akteure verschiedener Ebenen kommunizieren mittels Kanäle, die in der Regel durch ein Netzwerk realisiert sind. Einen Überblick über dieses Architekturmodell zeigt Abbildung 3–4. Dieser Ansatz orientiert sich insbesondere an der physikalischen Trennung von Zuständigkeiten, denn jeder der 3 Aspekte wird innerhalb kleinerer, physisch getrennter Teilsysteme auf der jeweiligen Ebene behandelt. Dieses Modell entstand als Konsequenz einer unvermeidbaren Notwendigkeit, Systemteile räumlich zu verteilen, da die Benutzer des Systems räumlich verteilt sind. Es folgt eine nähere Charakterisierung der einzelnen Ebenen:

- Präsentation

  Auf dieser Ebene wird das System dem Benutzer „präsentiert“. Es werden sowohl Aufträge an das System durch den Benutzer ermöglicht, als auch den Systemzustand in einer dem Benutzer angepaßten Form mitgeteilt. Die Benutzerinteraktionstools[2]

[1] Das Produkt R/3 der Firma SAP AG ist als Client/Server System auf der Basis der 3–Ebenen Architektur realisiert worden. Näheres erfährt man in [BMD94].

[2] Diese Tools werden auch „Thin Clients“ genannt, da sie nur für die Präsentation zuständig sind und somit eine kompakte Verhaltensbeschreibung besitzen.

sind lediglich in der Lage den Systemzustand von dem Anwendungsserver anzufordern und deren grafische Aufbereitung zu übernehmen. Das Verhalten dieser Tools kann jederzeit neuen Anforderungen im Bereich der (grafischen) Darstellung angepaßt werden, ohne die restlichen Systemteile verändern zu müssen. Sie haben nicht den hohen Ressourcenbedarf, der nötig wäre, wenn sie auch die geschäftslogischen Vorgänge abarbeiten müßten. In vielen anderen Systemarchitekturen wird die Präsentation und Geschäftslogik durch einen Akteur realisiert[1]. Dies kann bei großen Systemen zu vielen Probleme führen.

- Geschäftslogik

  Auf dieser Ebene werden die Geschäftsprozesse spezifiziert. Die Durchführung dieser im Vorfeld spezifizierten Vorgänge, übernehmen die Anwendungsserver. Die Anzahl der Anwendungsserver ist normalerweise sehr viel niedriger als die der Tools (in manchen Fällen wird sogar nur ein einziger Anwendungsserver benötigt). Alle Anwendungsserver eines Systems zeigen normalerweise gleiches Verhalten. Um die Skalierbarkeit und Fehlerredundanz zu erhöhen, wird diese Ebene jedoch meistens durch mehrere Anwendungsserver realisiert. Ein weiterer Vorteil besteht darin, daß Änderungen der Systemverhaltensbeschreibung (z.B. durch neue oder geänderte Geschäftsprozeßdefinitionen) nur ein Upgrade der Anwendungsserver erforderlich machen. Daraus folgt, daß das Gesamtsystem leichter zu warten, aber auch zu administrieren ist.

- Datenverwaltung

  Diese Ebene besteht in der Regel aus einem zentralen Datenbankverwaltungssystem, welches einzig und allein zentral alle Daten des Systems hält und verwaltet. Hierfür wird normalerweise ein sehr leistungsfähiger Rechner benötigt, um die große Anzahl der Anfragen der Anwendungsserver abarbeiten zu können. Diese Ebene kann jedoch auch als ein Verbund mehrerer, parallel arbeitender Datenverwalter realisiert werden. Diese Form ist insbesondere bei verteilten Systemen von hohem Interesse. Dabei werden die Daten von physisch verteilten Datenbanksystemen an unterschiedlichen Orten verwaltet. Die zentrale Datenverwaltung erlaubt normalerweise einfachere Mechanismen zur Synchronisation, Zugriffsverwaltung sowie Konsistenzerhaltung als die verteilte Datenverwaltung.

Die 3–Ebenen Architektur ist für die physische Trennung von Systemkomponenten großer Systeme gut geeignet. Ein solides Architekturkonzept sollte allerdings über dieses 3–Ebenen Modell hinaus weitere Zuständigkeitsbereiche behandeln. Als ein bewährtes Grundkonzept kann das 3–Ebenen Modell dennoch Hilfestellung für ein weitergehendes Architekturkonzept bieten. Zusätzliche Aspekte werden allerdings auf der Suche nach einem geeigneten Architekturkonzept für TimeLess behandelt werden müssen.

[1]Diese Tools nennt man oft auch „Fat Clients“, da sie sehr viel mehr an Ressourcen benötigen als die „Thin Clients“. Außerdem sind „Fat Clients“ schwerer zu administrieren, da eine Änderung der Geschäftslogik eine Neuinstallation des Clients notwendig macht.

3.2.5 Sicherheit

Insbesondere in Unternehmen spielt die Vertraulichkeit informationell verwalteter Daten eine große Rolle. Dabei muß man zwei Bereiche voneinander unterscheiden. Zum einen muß gewährleistet sein, daß Benutzer eines Informationssystems nur Zugang zu Daten erhalten, für die sie die Berechtigung besitzen. Zum anderen ist es wichtig kriminelle Angriffe wie z.B. Datenspionage in Folge des Abhörens von Kanälen zu verhindern bzw. zu erschweren.

- Berechtigungswesen

  Große Systeme, die von vielen Benutzern eingesetzt werden, benötigen einen Mechanismus, der nur bestimmten Benutzern Zugang zu bestimmten Daten ermöglicht, ein sogenanntes Berechtigungswesen. Dieser Mechanismus dient einerseits der Wahrung von Vertraulichkeit. Andererseits verhindert er auch Fehlbedienung bedingt durch Zugang zu Anwendungsbereichen, bzw. Nutzung des Systems in Rollen für die keine Erlaubnis vorliegt. Bei TimeLess muß z.B. verhindert werden, daß ein normaler Benutzer, also ein Leser, keine Regale umsortiert, denn dies ist einzig und allein Aufgabe des zuständigen Bibliothekars.

  Diesen Bereich in einer allgemeinen Form befriedigend zu lösen, stellt eine große Herausforderung an Softwaresysteme dar. Im Rahmen der TimeLess–Entwicklung soll ein Berechtigungskonzept gefunden werden, das jedoch nicht die Umsetzung des Berechtigungswesens selbst erfordert, sondern nur eine Möglichkeit vorsieht, jederzeit ein Berechtigungssystem nachträglich in TimeLess integrieren zu können, ohne architekturelle Konsequenzen befürchten zu müssen.

- Datenspionage, kriminelle Angriffe

  Insbesondere bei verteilten Systemen ist es wichtig Verfahren zu finden, die hohe Anzahl von Kommunikationskanälen möglichst abhörsicher zu machen. Hier ist es erforderlich durch geeignete Techniken, wie z.B. durch Verschlüsselung, die unterschiedlichen Protokollschichten gegen kriminelle Angriffe zu schützen (z.B. in Anlehnung an das ISO/OSI–Modell). Diese Überlegungen sollten jedoch nicht direkt in einem Architekturentwurf behandelt werden müssen. Beim Architekturentwurf kann man davon ausgehen, daß die Kommunikationskanäle als sicher realisiert werden können. Die Umsetzung dieser Sicherheit wird durch die Wahl geeigneter Technologien auf den Protokollschichten zur Realisierung der Kanäle sowie durch die Trägersysteme allgemein übernommen.

3.3 Anforderungen für das Abteilungsprodukt

Für die ersten Entwicklungsinkremente wurden einige Einschränkungen bezüglich der Funktionalität von TimeLess gemacht. Dabei steht in erster Linie die Architekturfindung und die darauf basierende Umsetzung und Implementierung von Konzepten im Vordergrund. Infolgedessen wird auf einige funktionelle Eigenschaften verzichtet, deren Realisierung nicht zu einem wesentlichen Erkenntnisgewinn führen würde. Es ist vorgesehen, diese Eigenschaften in

späteren Inkrementen hinzuzufügen. Dazu muß die Güte des Architekturkonzepts sehr hoch sein, um eine solche Erweiterbarkeit zu ermöglichen.

Sowohl in diesem Kapitel, als auch in den beiden Diplomarbeiten [Brand97] und [Langhauser97] wurden die im Vorfeld der Entwicklung bekannten Anforderungen und Eigenschaften des TimeLess–Produkts behandelt und spezifiziert. Nachfolgend werden kurz die Einschränkungen genannt, die in der ersten Phase aus den oben genannten Gründen gemacht werden. Diese reduzierte Funktionalität stellt die Basis des Abteilungsprodukts dar. Dennoch sollen die nachfolgend genannten Einschränkungen bei der Erarbeitung des Architekturkonzepts berücksichtigt werden.

3.3.1 Konkrete Einschränkungen

- Keine strukturierten Dokumente

  Die Strukturierung von Dokumenten stellt eine der Hauptsäulen der TimeLess–Philosophie dar. Dennoch wird zur Zeit noch aus diversen Gründen hierauf verzichtet. Zum einen setzt dieser Anforderungsbereich eine große Reife der Basiskomponenten (Navigationskomponente, ausgereifte Architektur) voraus. Zum anderen werden hierzu Editoren benötigt, deren Entwicklung zu diesem Zeitpunkt den Rahmen des Projekts sprengt. Weiterhin ist die momentane WWW[1]–Landschaft stark in Bewegung – insbesondere bei der Spezifikation neuer Normen bezüglich der Strukturierung von Dokumenten. Dabei scheint sich noch kein Standard etablieren zu können. Hier dürfte eine weitergehende Marktbeobachtung vorhandener Technologien zu einem späteren Zeitraum nützlich sein.

- Kein Büro.

  Das Büro unterscheidet sich vom restlichen Bibliotheksbereich ausschließlich dadurch, daß es nur für einen Benutzer zugänglich ist und grafisch etwas anders aussieht. Daher sollte das Büro nach Vorhandensein der unterschiedlichen Bibliothekskomponenten recht einfach implementiert werden können.

- Kein Katalog

  Die in [Brand97] dargestellte Katalogkomponente stellt zusammen mit der Suchmaschine ein Bereich dar, der recht klar von den sonstigen Anforderungen abgegrenzt werden kann. Bedingt durch die Unabhängigkeit der Komponenten, die zur Umsetzung dieser Funktionalität benötigt werden, kann zu diesem Zeitpunkt noch darauf verzichtet werden. Außerdem wird TimeLess im Rahmen des Abteilungsprodukts noch nicht die Größe erreichen, die eine Katalogkomponente zwingend erfordert. Sie würde in diesem Umfeld lediglich dem Komfort dienen.

[1] WWW ist Akronym und steht für den englischen Begriff „World Wide Web“ bzw. in deutsch: weltweites Netz. Hierbei spricht man von Internet–Technologien (wie z.B. dem Internet–Browser), welche die verschiedenen Dienste des Internet für den Endbenutzer zur Verfügung stellen.

- Ablagestruktur.

  Da TimeLess zunächst nur von einem kleinen Anwenderkreis benutzt wird, ist es derzeit noch nicht nötig die komplette Containerstruktur zu implementieren. Das TimeLess–Abteilungsprodukt beginnt somit mit dem Gebäude als Wurzelcontainer (siehe auch Abbildung 3–5). Dies vereinfacht die technische Implementierung ohne architekturelle Kompromisse eingehen zu müssen. Weiterhin wird derzeit auf den Containertyp „Sektion“ verzichtet. Dies stellt eine Prioritätenfestlegung zugunsten einer Arbeitsersparnis dar.

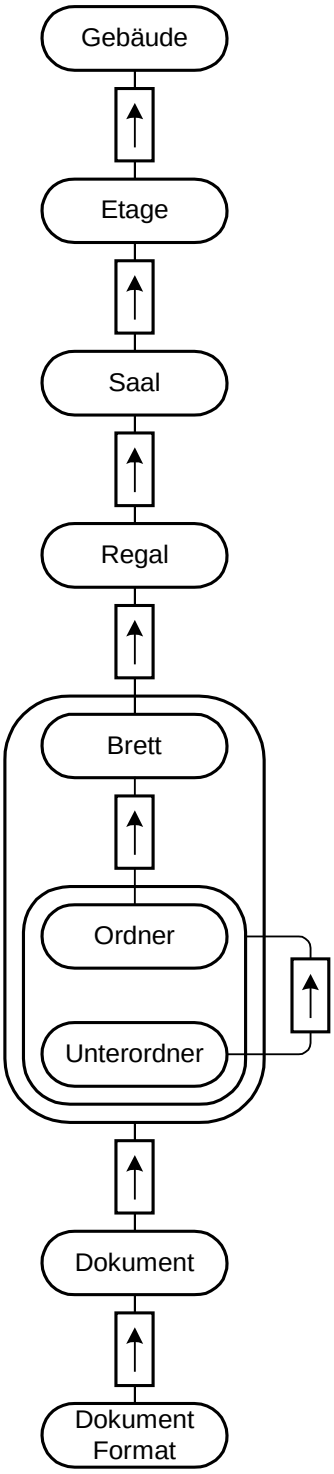

Abbildung 3–5: Die Ablagestruktur des Abteilungsprodukts

- Ergonomie besitzt keine Priorität

  Die subjektive Qualität insbesondere des Frontends, also der Benutzerschnittstelle, spielt eine große Rolle. Insbesondere bei der Vermarktung des TimeLess–Endprodukts ist die Güte der Benutzerschnittstelle ein entscheidender Faktor für den Erfolg von TimeLess. Beim Abteilungsprodukt ist es allerdings noch nicht von großer Bedeutung das schnellste, schönste oder ansprechendste Frontend zu entwickeln (in etwa entsprechend dem in [Brand97] entwickelten Demonstrationsprototyp). Das Ziel für das Abteilungsprodukt ist eine Benutzerschnittstelle, die es ermöglicht alle Operationen und Dienste von TimeLess vollständig nutzen zu können. Ergonomie sowie schnelle Antwortzeiten sollte erst dann eine Rolle spielen, wenn der dafür zu betreibende Aufwand nicht in Konflikt mit anderen wichtigeren Entwicklungsaufgaben steht.

- Dokumentformate

  Um zu ermöglichen, daß jedem TimeLess–Dokument mehrere technische Dateiformate zugeordnet werden können, wurde als Erweiterung der in [Brand97] und [Langhauser97] spezifizierten Ablagestrukturen eine neue Entität eingeführt. Mehrere dieser Dokumentformat–Entitäten können einem logischen Dokument zugeordnet sein (siehe auch Abbildung 3–5). So kann ein man beispielsweise einen Aufbauplan in mehreren technischen Formaten ablegen. Dazu können z.B. das mit einem grafischen Autorentool erstellte Ursprungsformat gehören, als auch zusätzlich weitere, für andere Anwendungszwecke besser geeignete Formate. Dokumente sind somit logische Behälter mehrerer technischer Formate, die in unterschiedlichen Formen die gleiche logische Entität darstellen.

  Vorrangig soll beim Einsatz verschiedener technischer Formate auf browserkompatible WWW–Dokumentformate[1] zurückgegriffen werden. Dadurch kann man das Problem umgehen, für jedes Format einen Viewer[2] besitzen zu müssen. Da die Mächtigkeit dieser WWW–Formate jedoch relativ beschränkt ist, werden zusätzlich noch bestimmte andere Formate zugelassen. Diese werden auf die in der Laborumgebung üblichen Formate beschränkt.

[1] Damit sind Dateiformate gemeint, die jeder marktübliche Internetbrowser interpretieren und darstellen kann, wie z.B. Hypertext Meta Language (.html), Graphic Interlaced Format (.gif) und Joint Photographic Experts Group (.jpg)

[2] Viewer können den Inhalt von technischen Dokumentformaten nur interpretieren und darstellen. Darüber hinausgehende Änderungen an Dokumentformaten vornehmen zu können, erfordert spezielle Editoren.

Kapitel 4 Das Unternehmensmodell

Eine der wohl schwierigsten Aufgaben im Rahmen des TimeLess–Entwicklungsprozesses stellt sicherlich der Entwurf einer geeigneten Systemarchitektur dar. Die Komplexität der Aufgabenstellung sowie die erhoffte Effektivität der nachfolgenden Entwicklungsiterationen erfordert eine möglichst natürliche Vorgehensweise, die man durch ein realitätsnahes Modell dokumentieren kann. Ein solches Modell wird an dieser Stelle vorgestellt um stets einen Bezug zum nachfolgenden Architekturentwurf herstellen zu können. Diese Vorgehensweise soll ein besseres Verständnis ermöglichen.

4.1 Das Informationsversandhaus als Modell

Bereits im vorherigen Kapitel konnte gezeigt werden, wie Unternehmensstrukturen und –vorgänge sehr gut zur Modellierung technischer Zusammenhänge geeignet sein können. Auch im folgenden Modell soll der Aufbau eines Großunternehmens die Hinführung zur eigentlichen TimeLess–Architektur erleichtern. Es handelt sich dabei um ein Informations- und Warenversandhaus. Dieses vertreibt Waren in Form von Multimediadokumenten wie Bücher, Videos und CDs aber auch Information über diese Waren, wie z.B. deren Autor, Titel oder die Anzahl Kunden, die sich bisher für dieses Dokument interessiert haben. Man könnte sagen, daß man als Kunde nicht nur Waren kaufen kann sondern auch Information über diese Waren.

Abbildung 4–1 zeigt die grobe Struktur eines solchen Unternehmens. Dieses möchte möglichst flexibel sein, um sich an ändernde Kundenwünsche möglichst schnell anpassen zu können. Hierzu hat es mehrere Auftragsabwicklungszentren global verteilt. Diese erhalten Aufträge von den diversen Kundenbetreuern, die beim Kunden vor Ort tätig sind. Diese Kundenbetreuer sind dafür verantwortlich, die Wünsche und Aufträge des Kunden fachgerecht an die für diesen Kunden zuständige Abteilung des Auftragsabwicklungszentrums weiterzuleiten. Ebenso werden Auftragsergebnisse durch den Kundenbetreuer an seinen Kunden überreicht. Der Kunde selbst muß lediglich vom Kundenbetreuer bereitgestellte Formulare ausfüllen und an diesen überreichen. Man kann sich das so vorstellen, daß der Kunde Aufträge durch das Ausfüllen von Formularen formuliert. Diese übergibt er an seinen Kundenbetreuer. Dieser interpretiert den Kundenauftrag, wandelt ihn in ein dem Informationsversandhaus bekannten

Format um, und versendet den Auftrag mit dem zur Verfügung stehenden Vermittlungsdienst (das könnten verschiedene sein, z.B. per Telefax, Telefon oder Kurier).

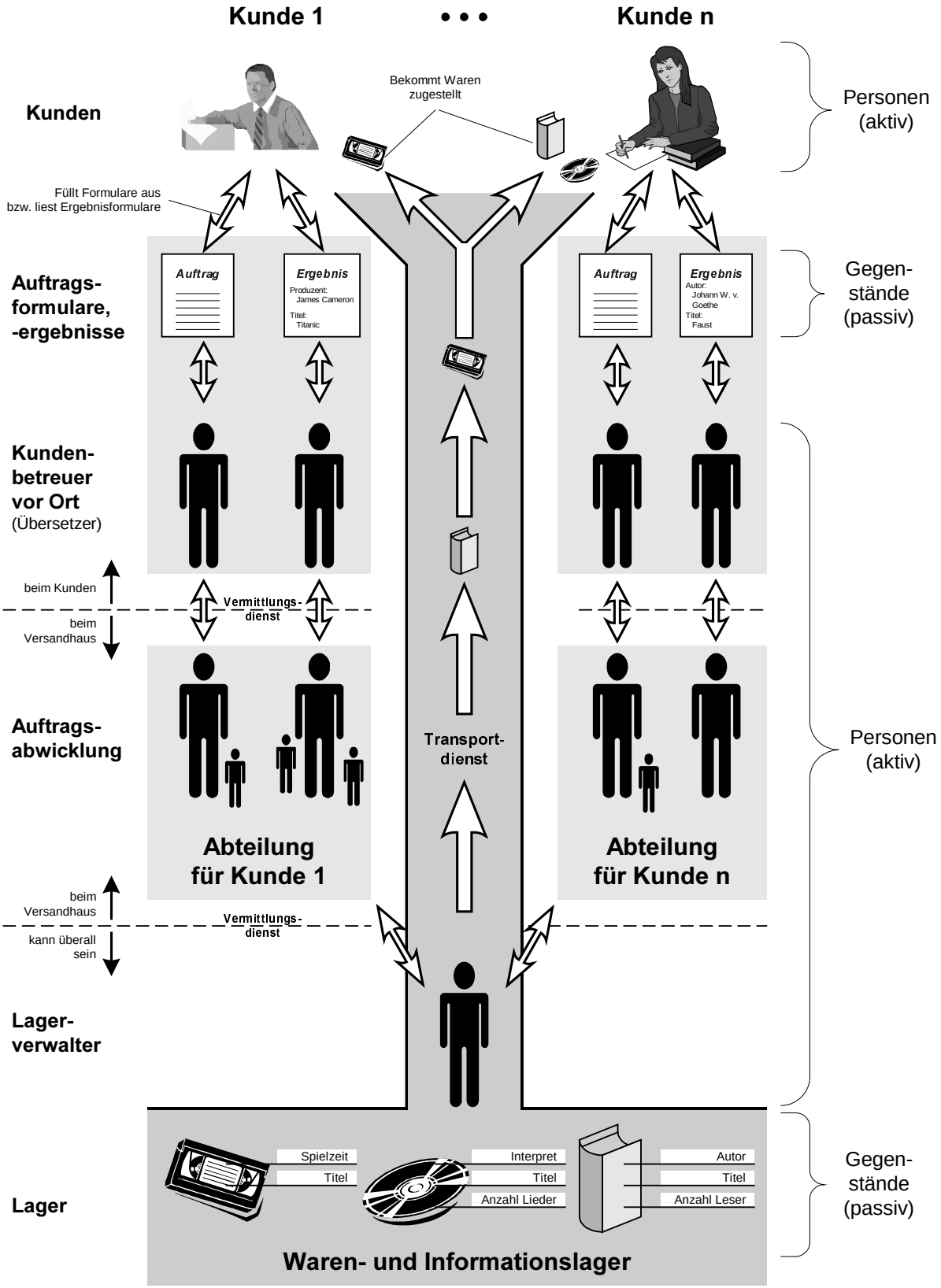

Abbildung 4–1: Das Informationsversandhaus als Modell

Der Auftrag wird nun von der Auftragsabwicklungsabteilung angenommen und bearbeitet. An dieser Stelle müssen verschiedene Prüfungen gemacht werden. So muß z.B. kontrolliert werden, ob der Kunde die Berechtigung besitzt einen solchen Auftrag abzugeben (z.B. weil noch offene Rechnungen ausstehen). Ebenfalls muß geprüft werden, ob der Auftrag überhaupt durchführbar ist (z.B. ob die gewünschte Menge noch zur Verfügung steht). Sollte der Auftrag durchführbar sein, muß noch festgestellt werden, ob es sich um eine Waren- oder eine Informationsbestellung handelt. Bei einer Warenbestellung wird der Auftrag an den Lagerverwalter weitergegeben, der die gewünschten Waren aus dem Lager besorgt und dem Kunden mittels Transportdienst zustellen läßt. Handelt es sich hingegen um eine Informationsbestellung, dann wird bei Bedarf der Lagerverwalter nur nach der benötigten Information befragt. Diese wird dem Kunden wiederum mittels Kundenbetreuer auf einem Ergebnisformular mitgeteilt.

Der bisher beschriebene Auftragsvorgang stellt nur einen kleinen Ausschnitt der verschiedenen Aufgaben innerhalb des Versandhauses dar. Die Organisation dieses Unternehmens besitzt noch weitere Eigenschaften und Möglichkeiten, die näher erörtert werden sollen. Insbesondere werden die verschiedenen Dienste und Zuständigkeiten näher beschrieben, und deren Notwendigkeit begründet. Ziel dieser Darstellungen ist es, die in enger Analogie auch in TimeLess vorkommenden Architekturmerkmale besser zu verstehen.

4.1.1 Die Personen des Informationsversandhauses

- Der Kunde

  Kunden möchten Aufträge in einer intuitiven, benutzerfreundlichen Art erteilen können. Hierbei sind Mißverständnisse bedingt durch unklare Formulierungen unerwünscht. Daher müssen die Formulare, die der Kunde ausfüllt, mit möglichst wenig Einarbeitungszeit benutzt werden können. Sie müssen sich durch ihre Darstellung fast von selbst dokumentieren und erklären.

  Außerdem ist der Kunde nicht an Versandhausinterna interessiert. Die Vorgänge, die erforderlich sind um seinen Auftrag abzuarbeiten, sind für den Kunden nicht von Bedeutung. Der Kunde kennt nur seine Kundenbetreuer und die Formulare, die er diesen überreicht. Dadurch kann er sich voll und ganz auf seine eigentliche Tätigkeit konzentrieren.

  Ausgefüllte Formulare werden vom Kunden direkt an den Kundenbetreuer überreicht. Dieser steht vor Ort beim Kunden persönlich zur Verfügung, um die Formulare mitzunehmen (Formulare werden nicht per Kurier oder Vermittlung an den Kundenbetreuer überreicht). Ebenfalls kann der Kundenbetreuer Hilfestellung für fehlerhaft ausgefüllte Formulare vor Ort leisten.

- Der Kundenbetreuer

  Kundenbetreuer haben im wesentlichen zwei Aufgaben:

  1) Sie müssen den Inhalt der durch den Kunden überreichten Formulare in ein unternehmensweit standardisiertes Format umwandeln. Als Beispiel kann man sich hier vorstellen, daß es verschiedensprachige Formulare gibt, wie deutsche und japanische Formulare. Die Unternehmenssprache ist jedoch Englisch. In diesem Fall muß der Kundenbetreuer eine Übersetzung der jeweiligen Kundensprache ins Englische anfertigen, bevor er den Auftrag an die Auftragsabwicklung weiterleitet.

  2) Kundenbetreuer sind dafür zuständig, den „ortsüblichen" Transportdienst zu benutzen. Das könnte z.B. bedeuten, daß sie zur Auftragsweiterleitung das Telefon benutzen. In anderen Fällen könnte aber auch ein Telefax oder ein Kurierdienst benutzt werden. Dadurch müssen Kunden nicht wissen, wie ihre Aufträge weitergeleitet werden. Folglich müssen die Formulare, die sie ausfüllen, nicht für den Versand oder die Weiterleitung vorbereitet sein. Allerdings muß der vom Kundenbetreuer weitergeleitete Auftrag entsprechend verpackt sein. Beispielsweise müssen bestimmte Umschläge beim Einsatz eines Kurierdiensts benutzt werden. Beim Telefax muß der Auftrag auf einer A4–Seite mit Adressat stehen usw.

- Die Auftragsabwicklung

  In der Auftragsabwicklung steht für jeden Kunden eine eigene Abteilung von Mitarbeitern bereit. An dieser Stelle erfolgt ein Bruch zur Realität, denn kein Unternehmen würde für jeden Kunden eigens eine ganze Abteilung bestehend aus mehreren Mitarbeitern einstellen. Dennoch wäre das auch in der Realität wünschenswert, da sich dadurch jeder Mitarbeiter voll und ganz auf seine Aufgaben bezüglich dieses einen Kunden konzentrieren könnte. Dadurch werden Mitarbeiter nicht überfordert durch zu viele Zuständigkeiten.

  Jede Abteilung besitzt viele Mitarbeiter, die bei Bedarf jederzeit um zusätzliche Mitarbeiter erweitert werden kann. Hierfür gibt es wiederum eine Personalabteilung, die jedoch zur Vereinfachung nicht in Abbildung 4–1 dargestellt ist. Zusätzliche Mitarbeiter werden dann benötigt, wenn die vorhandenen beschäftigt sind, oder sich bereits auf bestimmte Aufträge spezialisiert haben.

  Aufträge werden von dem Vermittlungsdienst direkt an den zuständigen Mitarbeiter der entsprechenden Abteilung verteilt. Nach deren Annahme erfolgen zwei Formen von Prüfungen: eine juristische sowie eine operationelle Prüfung. Bei der ersten wird geprüft, ob der Kunde den vorliegenden Auftrag ausführen darf, d.h. ob er die Berechtigung besitzt einen solchen Auftrag abzusetzen. Die zweite, operationelle Prüfung, prüft die Durchführbarkeit des Auftragsvorgangs durch das Versandhaus. Das bedeutet, daß festgestellt wird, ob der Auftrag zum aktuellen Zeitpunkt unter den aktuellen Umständen möglich ist.

  Sollten die Prüfungen erfolgreich sein, werden die gewünschten Informationen vom Informationslager besorgt. Wenn es sich um einen Warenauftrag handelt, so wird der Versand der Waren an den Kunden in Auftrag gegeben. Für diese Aufgaben zustän-

dig ist der ebenfalls in Abbildung 4–1 abgebildete Lagerverwalter. Dieser gibt zum einen Auskunft über von ihm verwaltete Waren. Zum anderen koordiniert er den Versand von Waren an den Kunden.

Es kann auch vorkommen, daß die von der Auftragsabwicklung benötigte Information nicht vom Lagerverwalter besorgt werden muß, sondern bereits vorhanden ist. Wenn Information, wie beispielsweise der Autor eines bestimmten Buchtitels, vom Kunden angefordert wird, dann kann es sein, daß der Mitarbeiter der Abteilung diese Information bereits kennt. Ein Mitarbeiter, der eine solche Anfrage bereits bearbeitet hat, ist in der Lage sich das Ergebnis zu „merken“. Er kann sich aber auch Informationen merken, in dem er vom Lagerverwalter mehr anfordert als unbedingt für die Durchführung des Auftrags benötigt wird. Wenn beispielsweise vom Kunden ein Buch bestellt wird, dann kann sich der zuständige Mitarbeiter alle verfügbare Information zu diesem Buch zukommen lassen obwohl nur das Buch selbst angefordert wurde. So könnte er beispielsweise auch erfahren, wer das Buch bereits gelesen hat. Sollte der Kunde in einem späteren Auftrag diese Information anfordern, muß diese nicht angefordert werden, sondern sie kann ihm direkt durch den Auftragsbearbeiter zur Verfügung gestellt werden.

Diese Vorgehensweise läßt sich nicht für Waren, wie Bücher oder Videos einsetzen, da es sich hierbei um physisch vorhandene Entitäten handelt, die zu einem bestimmten Zeitpunkt nur einen Besitzer haben können. Es nicht möglich sich zum Beispiel ein Buch zu „merken“. Entweder man ist in dessen Besitz oder nicht. Das einzige was man sich merken kann, sind Informationen zum Buch, wie sein Autor, Titel aber auch Inhalt. Ebenso kann man sich die Identität merken, d.h. das Wissen um seine Existenz.

Das Unternehmen gruppiert mehrere dieser kundenspezifischen Auftragsabteilungen in regionalen Großzentren zusammen. Dadurch ist es möglich, daß bestimmte Unternehmensdienste abteilungsübergreifend von den Großzentren zur Verfügung gestellt werden können. So können sich die verschiedenen Abteilungen beispielsweise eine Personalabteilung teilen. Ebenso können Transportdienste gemeinsam benutzt werden. Dadurch kann das Unternehmen eine bessere Auslastung seiner Ressourcen erzielen. Weiterhin können Schwankungen der Kundenzahl durch eine geschickte Verteilung auf die verschiedenen Zentren besser bewältigt werden.

- Der Informationsverwalter (Lagerverwalter)

  In Abbildung 4–1 ist lediglich ein einziger Lagerverwalter zu erkennen. Diese Abbildung geht davon aus, daß der Auftragsbearbeiter bereits den zuständigen Lagerverwalter sowie seinen Ort kennt. Da nicht unbedingt eine zentrale Lagerverwaltung vorausgesetzt wird, muß es jemanden geben, der den aktuellen Ort aller Lagerverwalter kennt. Diese Aufgabe übernimmt die nicht abgebildete Unternehmensauskunft. Man kann diesen Dienst in etwa mit der Telefonauskunft vergleichen (daher muß er auch immer verfügbar sein). Ein Mitarbeiter der Auftragsabwicklung möchte eine Informationseinheit besorgen (die er identifizieren muß). Er kennt die Existenz der Informationseinheit, nicht jedoch ihren Lagerort, bzw. –verwalter. Hierzu ruft er

die Unternehmensauskunft an, die ihm den Ort des zuständigen Lagerverwalters mitteilt. Also der gleiche Vorgang, der benötigt wird, um die Telefonnummer einer bekannten Person mittels Telefonauskunft in Erfahrung zu bringen.

Diese Auskunft wird natürlich nur benötigt, falls die Identifizierbarkeit einer gewünschten Entität nicht auch deren Lagerort impliziert[1]. Da in diesem Fall durch bloße Kenntnis der Existenz einer gelagerten Entität, der Lagerort, bzw. –verwalter nicht bekannt sein kann, ist das Vorhandensein einer solchen Auskunft sehr wichtig.

Abbildung 4–1 zeigt nur einen Lagerverwalter. Dieser bietet dem Auftragsbearbeiter die Möglichkeit über die von ihm verwalteten Informationen und Waren Auskunft zu geben. Über einen Vermittlungsdienst, wie z.B. per Telefon, werden diese Auskünfte durch den Lagerverwalter erteilt. Sollte der Auftragsbearbeiter keine Auskunft benötigen, sondern eine Auslieferung bestimmter Waren an den Kunden beauftragen wollen, so kann er dies dem Lagerverwalter mitteilen. Der Auftragsbearbeiter muß dabei die gewünschte Ware identifizieren können und den Lieferort des Kunden mitteilen. Danach kann der Transport bzw. Versand der Waren vom Lagerverwalter veranlaßt werden. Ein Buch, welches durch den Kunden bestellt wird, wird vom Lager direkt zu ihm transportiert/versandt.

4.1.2 Die Gegenstände des Informationsversandhauses

Wie in Abbildung 4–1 erkennbar, werden sowohl die vom Kunden auszufüllenden Formulare als auch die gelagerten Informationseinheiten als passive Gegenstände dargestellt. Diese sind, im Gegensatz zu den aktiven Personen, nicht in der Lage selbst zu handeln. So kann ein Formular beispielsweise nicht mit einem Kunden reden oder mit ihm interagieren. Vereinfacht ausgedrückt, mit Gegenständen wird etwas gemacht, und zwar grundsätzlich von Personen.

- Das Auftragsformular bzw. –ergebnis

  Das Auftragsformular dient der Auftragsformulierung seitens des Kunden. Da diese Formulierung für den Kunden einfach und intuitiv sein soll, ist es wichtig, Formulare zu benutzen, die der Kunde leicht und schnell versteht. Außerdem können diese Formulare jederzeit ausgetauscht werden oder in anderer Art und Weise gestaltet werden, um jedem Kunden das für ihn geeignete Formular zur Verfügung zu stellen.

  Formulare wird mit Hilfe der entsprechenden Kundenbetreuer ausgefüllt. Hier kann es vorkommen, daß der Kundenbetreuer merkt, daß das Formular nicht ordnungsgemäß ausgefüllt wird. Dies kann er dem Kunden mitteilen, was zu einer Vereinfachung der Auftragserstellung führt. Ein inhaltlich korrekt ausgefülltes Formular wird nun, wie bereits erläutert, vom Kundenbetreuer angenommen. Dieser benutzt das Formular lediglich, um seinen Auftrag an die Auftragsabwicklung zu formulieren und weiterzuleiten.

[1] Der Unterschied zwischen Identifikation und Zeigbarkeit wird im Anhang A Identifizieren und Zeigen an einem konkreten Beipiel des TimeLess–Systems gezeigt.

Sollte der Kunde Informationen anfordern, nicht Waren, dann ist für ihn die physische Form der Information nicht von Bedeutung. Vielmehr interessiert ihn der informationelle Gehalt der Antwort des Auftrags. Die Beantwortung solcher Aufträge zur Informationsbeschaffung erfolgt mittels eines Ergebnisformulars, welches der Kundenbetreuer für den Kunden ausfüllt. Möchte der Kunde beispielsweise wissen, wer der Autor eines bestimmten Buchs ist, so trägt der Kundenbetreuer diese Information, die er von der Auftragsabteilung erhalten hat, in einem dafür ausgelegten Formular ein. Dieses Ergebnisformular überreicht er dem Kunden. Somit erfolgt die Kommunikation zwischen Kunden und Kundenbetreuer mittels Formulare.

- Die gelagerte Information

  Bei der Information, die vom Versandhaus angeboten wird und in örtlich beliebig verteilten Lagern aufbewahrt wird, gibt es zwei verschiedene Typen von Information.

  Zum einen gibt es Waren, die zwar vom Versandhaus identifiziert werden können, deren genauer Inhalt jedoch unbekannt ist. So ist das Versandhaus lediglich an der Lagerung eines Buchs oder eines Videos interessiert, nicht jedoch an deren Inhalt (sofern dies nicht zum Informationsangebot gehört). Bei Waren ist nicht der informationelle Gehalt von Bedeutung, sondern vielmehr deren physische Attribute, wie Größe und Gewicht. Der Kunde wird derjenige sein, der den Inhalt dieser Waren interpretieren und nutzen kann.

  Der zweite Typ besteht aus den gelagerten Informationen, bei denen nicht die physischen Eigenschaften, sondern die informationelle Form von Bedeutung ist. Dies können Informationen über gelagerte Waren sein, wie z.B. Titel und Autor eines Buchs. Es können aber auch Informationen sein, die nichts mit Waren zu tun haben. Eine Liste aller in Deutschland befindlichen Hochschulen und deren Fachbereiche könnte das Versandhaus beispielsweise als reine Information anbieten.

  Die Unterscheidung dieser beiden Informationstypen ist für das Versandhaus wichtig, denn Waren müssen sowohl anders gelagert werden als die sonstigen Informationen. Ebenso werden Waren auch transportiert im Gegensatz zu Informationen, die vermittelt werden. Da die physische Form von Informationen für den Kunden nicht von Bedeutung ist, kann das Versandhaus Mechanismen vorsehen, diesen zweiten Typ möglichst effizient zu lagern und zu vermitteln (z.B. per Telefon). Weiterhin kann von diesem zweiten Typ jederzeit eine Kopie gemacht werden, was bei Waren nicht möglich ist. Waren müssen physisch als Ganzes transportiert werden, also kommen nur langsame Transportdienste wie z.B. die Post in Frage.

  Durch eine Klassifizierung von Waren können bestimmte Informationen standardisiert verwaltet werden. So kann das Versandhaus für jede Warenklasse ein Formular erarbeiten, welches die charakteristischen Attribute eines Warenexemplars verwaltet (z.B. Titel und Autor von Büchern). Somit werden automatisch wichtige Attribute zu bestimmten Warenklassen mitverwaltet. Dadurch kann dem Kunden zu bestimmten Waren eine standardisierte Menge von Attributen als Information zur Verfügung gestellt werden.

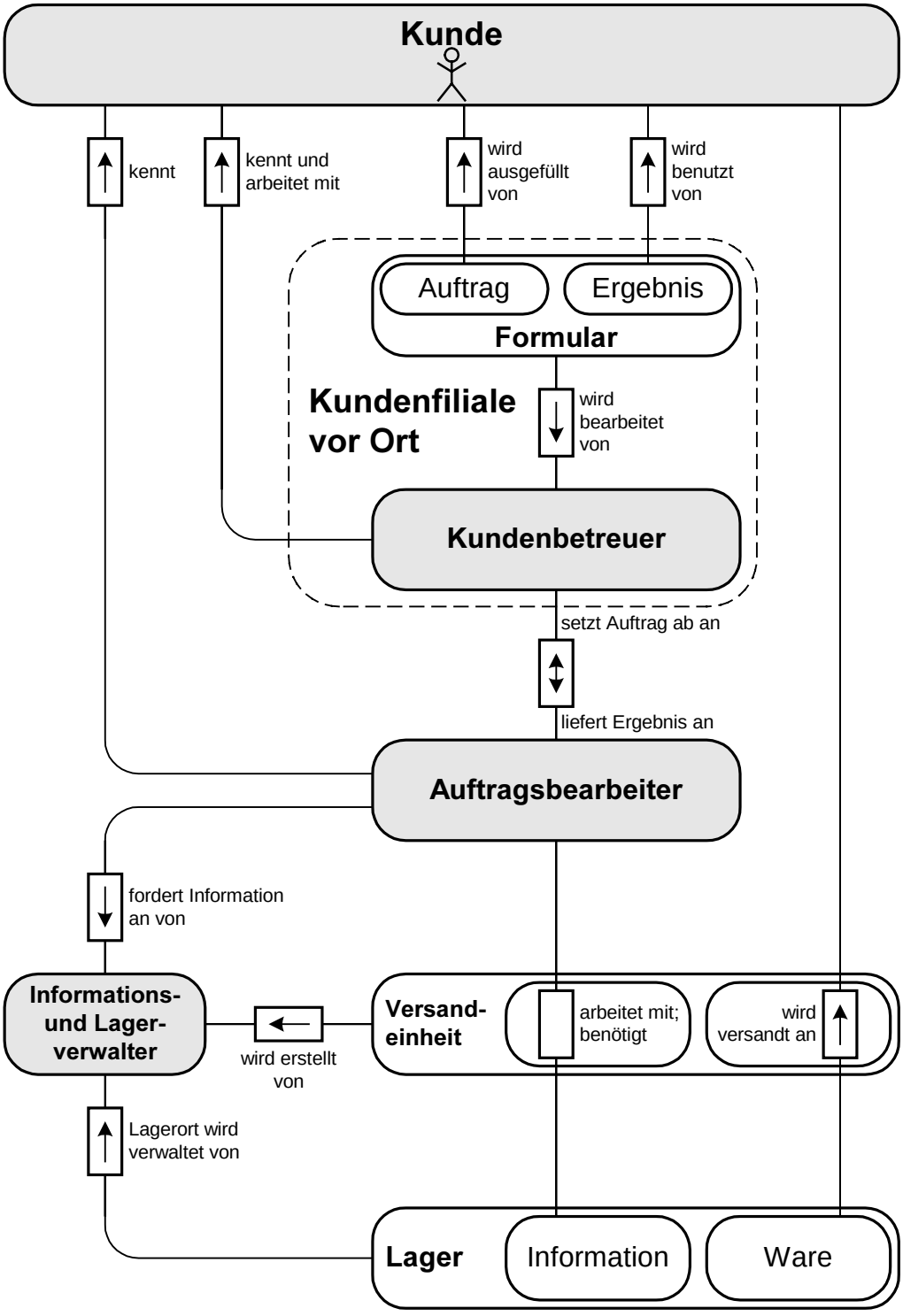

Abbildung 4–2: Beziehungen zwischen Komponenten des Informationsversandhauses

4.1.3 Beziehungen des Versandhauses

Das bisher vorgestellte Modell beschreibt grob den Aufbau eines Versandhauses. Insbesondere wurden bisher die Aufgaben der jeweiligen Versandhauskomponenten, differenziert nach

aktiven und passiven Komponenten, beleuchtet. Daraus gingen zum Teil implizit die Beziehungen zwischen Komponenten hervor. Diese Beziehungen werden im E/R–Diagramm in Abbildung 4–2 präzisiert. Insbesondere die Entitäten, die zum Versand von Waren bzw. Vermittlung von Information benötigt werden, werden in diesem Bild konkret dargestellt. Hierauf wurde bisher aus Gründen des leichteren Verständnisses verzichtet. Die neu hinzugekommenen, bisher nicht erwähnten Aspekte im Bild sollen nachfolgend näher beleuchtet werden. Die im Bild grau dargestellten Entitäten stellen die aktiven Komponenten dar, die Weißen die Passiven. Auf Kardinalitätsbeziehungen wurde noch verzichtet, da es an dieser Stelle nicht zur Erkenntnisgewinnung beitragen würde.

Sowohl Kundenbetreuer als auch Auftragsbearbeiter kennen genau einen Kunden, für den sie bestimmte Dienste erbringen. Die „kennt“–Beziehungen, die in Abbildung 4–2 zwischen diesen Entitäten zu sehen ist, ist allerdings nur eine implizite Beziehung, denn sowohl der Auftragsbearbeiter als auch der Kundenbetreuer werden nur für diesen Kunden eingestellt und können daher von demjenigen, der sie eingestellt hat jederzeit Informationen zum Kunden erfahren. Die Entitäten, die für diese Einstellungen zuständig sind, werden bei der Schaffung von Komponenten im folgenden Abschnitt näher erläutert.

Es ist wichtig, daß sowohl der Kundenbetreuer als auch der Auftragsbearbeiter ihren Kunden kennen. Der Kundenbetreuer muß ihn kennen, da er vor Ort beim Kunden tätig ist und mittels Formulare mit ihm kommuniziert. Der Auftragsbearbeiter muß ihn kennen, damit er prüfen kann, ob dieser Kunde die gewünschten Aufträge absetzen darf (juristische Prüfung), aber auch um den Versand von Waren an seinen Kunden zu koordinieren. Dieser Aspekt wurde bisher nicht näher betrachtet. Stellt der Auftragsbearbeiter fest, daß die ihm zur Verfügung stehenden Informationen nicht genügen, um einen Auftrag abzuwickeln, so muß er die Informationen vom Informations– und Lagerverwalter anfordern. Dieser kennt den Lagerort, an dem die Information abgelegt ist. Die an diesem Ort gelagerte Information muß nun dem Auftragsbearbeiter zur Verfügung gestellt werden. Dies setzt allerdings wegen der räumlichen Trennung eine Vermittlung bzw. einen Transport der Information an den Auftragsbearbeiter voraus. Diese Vermittlung wird durch die Versandeinheit, die vom Informations– und Lagerverwalter veranlaßt wird, realisiert. Zusätzliche Information, die zur Vermittlung benötigt wird (wie z.B. Information für den nicht dargestellten Vermittlungsdienst oder Auftragsnummer usw.), wird an diese Versandeinheit angehängt[1]. Versandeinheiten können z.B. Telefaxseites mit der geforderten Information sein.

Eine ähnliche Vorgehensweise findet beim Versand von Waren statt. Da der Inhalt dieser Waren, wie bereits geschildert, nicht von Interesse für den Auftragsbearbeiter ist, wird veranlaßt, daß Waren direkt an den Kunden versandt werden (wiederum vom Informations– und Lagerverwalter). Auch hier wird die Zuordnung der zu versendenden Ware zu dem empfangenden Kunden als Relation dargestellt. Aus dieser Relation wird ebenfalls durch Objektifizierung eine Transportentität geschaffen um Informationen, die nur zum Transport benötigt werden, zu verwalten.

[1] Die Attributierung einer Relation nennt man auch Objektifizierung.

4.1.4 Strukturvarianz im Versandhaus

Das E/R–Diagramm in Abbildung 4–2 dient als Darstellung der Beziehungen im strukturinvarianten Fall, also dann, wenn die Menge der aktiven und passiven Komponenten konstant bleibt. Daß dies jedoch nicht der Normalfall sein wird, läßt sich an einem kleinen Beispiel erläutern. Ein Versandunternehmen wird wachsen wollen, um seinen wirtschaftlichen Erfolg zu sichern. Dies kann es nur, indem mehr Kunden gewonnen werden können oder indem höherwertigere Dienste angeboten werden. Es wurde bei der Einführung in dieses Modell bereits angedeutet, daß Auftragsbearbeiter in Abteilungen zusammengefaßt werden. Jede Abteilung ist genau für einen Kunden zuständig. Mehrere Abteilungen wiederum werden innerhalb von Großzentren zusammengefaßt, die zentrale Dienste zur Verfügung stellen (Hausverwaltung, Anmeldung). Diese Aufgaben dienen primär der Verwaltung der sich stets ändernden Struktur des Unternehmens, hauptsächlich bedingt durch das ständige Hinzukommen und Wegfallen von Kunden, aber auch bedingt durch Wachstum oder Verkleinerung des Unternehmens.

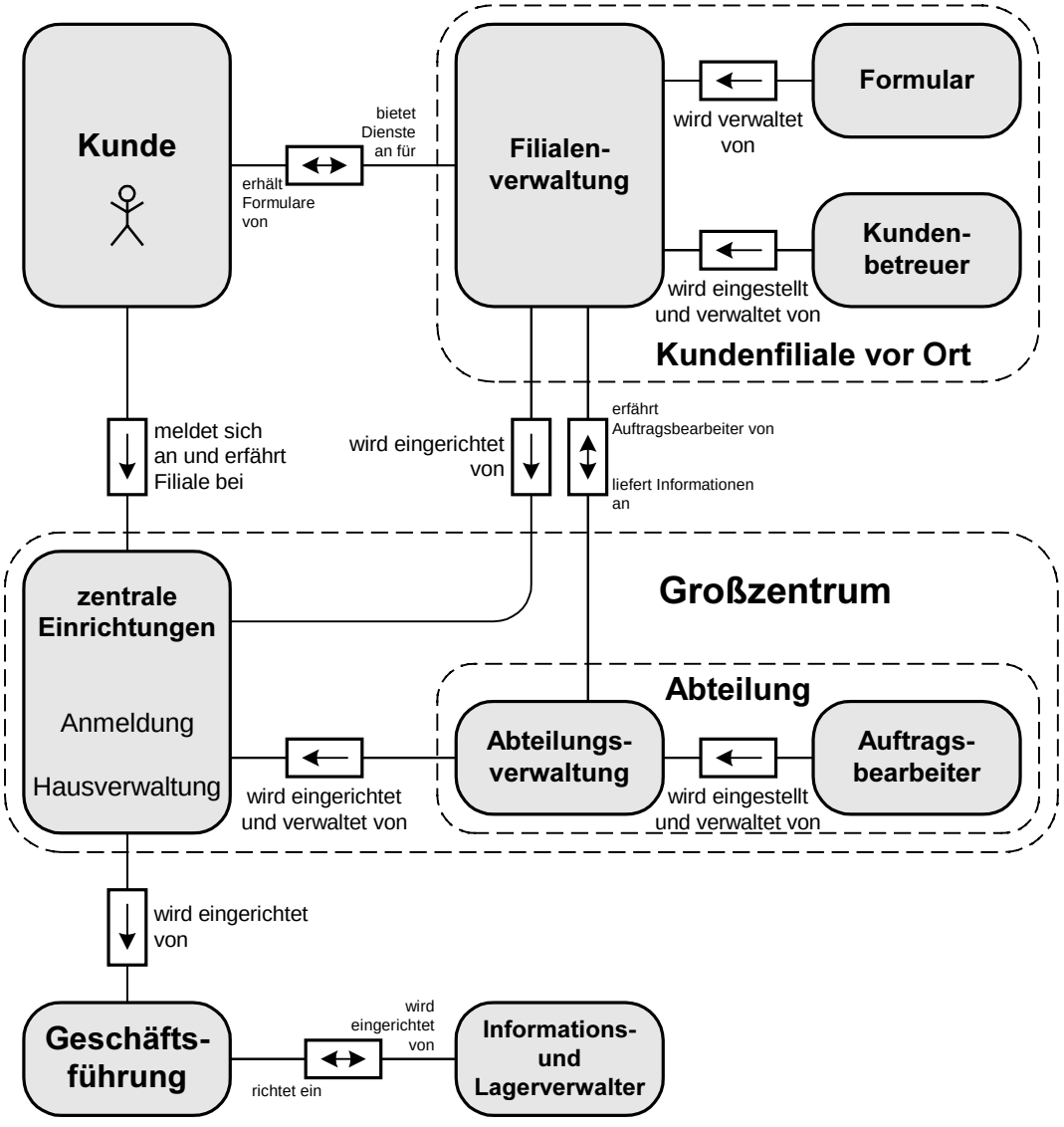

Abbildung 4–3: strukturelle Veränderungen im Versandhaus

Die Entitäten innerhalb des Unternehmens, die für diesen neu hinzugekommenen Aspekt benötigt werden, sind im E/R–Modell in Abbildung 4–3 zu sehen. Um das Unternehmen aufzubauen, bedarf es der Geschäftsführung, die Grundstrukturen schafft. Zu diesen Grundstrukturen, ohne die das Versandhaus keine Geschäftstätigkeit aufnehmen könnte, gehört der Informations– und Lagerverwalter sowie die zentralen Einrichtungen des ersten Großzentrums. Im Laufe der Zeit können weitere Großzentren benötigt werden, die dann ebenfalls von der Geschäftsführung eingerichtet werden (z.B. bei Unternehmensexpansion auf neuen Märkten). Zur zentralen Einrichtung eines Großzentrums gehören in diesem Fall die Hausverwaltung sowie die Anmeldung.

Die zentralen Einrichtungen sind dafür zuständig, neue Räume und Infrastruktur für die Abteilungen zur Verfügung zu stellen und die Abteilungsverwaltung darin einzurichten. Dabei teilt sie dem Abteilungsverwalter den Kunden mit, für den diese Abteilung zuständig sein wird (siehe Abbildung 4–2). Die Abteilungsverwaltung stellt bei Bedarf neue Mitarbeiter als Auftragsbearbeiter ein und übernimmt deren Betreuung. Wenn eine Abteilung geschaffen ist, kann sie eigenständig für den ihr zugewiesenen Kunden operieren.

Damit eine Zuteilung des Kunden möglich wird, muß sich der Kunde zuerst anmelden. Dazu muß der Kunde wissen, bei wem er anrufen muß, um den Anmeldevorgang einzuleiten. Jedes Großzentrum verfügt über eine solche Anmeldung. Daher muß der Kunde sich für ein Großzentrum entscheiden. Angenommen es gäbe Großzentren in Frankfurt, Hamburg und Paris. Vermutlich wird der Kunde sich mit dem ihm am nächsten befindlichen Großzentrum in Verbindung setzen. Also wenn er in Walldorf zu Hause ist, wird er sich wahrscheinlich in Frankfurt anmelden. Den Zugang zur Anmeldung kann er über unterschiedlichste Quellen erfahren haben. Beispielsweise hat er die Telefonnummer aus der Werbung. Oder er erhielt sie von einem anderen Kunden. Bei der Erstanmeldung werden seine Kundendaten für spätere Wiederverwendung erfaßt.

Bei der Anmeldung erfolgt zuerst die bereits besprochene Einrichtung der Abteilung. Danach wird vor Ort beim Kunden die Einrichtung einer Filiale veranlaßt, vertreten anfangs durch die Filialenverwaltung. Der Begriff Filiale wurde hier gewählt, da er im Unternehmen gängig ist, allerdings muß man sich vorstellen, daß diese Filiale nur für diesen einen Kunden direkt bei ihm vor Ort eingerichtet wird. Bei Einrichtung wird einer Filiale mitgeteilt, welche Abteilung für die Auftragsabwicklung der Filiale zur Verfügung steht, als auch für welchen Kunden sie zuständig sein wird (Diese Zusammenhänge sieht man auch in Abbildung 4–2). Hier ist es wichtig, daß die Filialenverwaltung die Abteilungsverwaltung kennt, da sie von ihr erfährt welche Auftragsbetreuer für die jeweiligen Aufträge zuständig sind.

Nachdem die Filialenverwaltung eingerichtet wurde, ist sie in der Lage dem Kunden ihre Dienste zur Verfügung zu stellen (wie das Ausfüllen von Auftragsformularen). Fertig ausgefüllte Formulare werden an die dafür von der Filialenverwaltung eingestellten Kundenbetreuer gegeben. Jedem dieser Kundenbetreuer wird genau ein Auftragsbetreuer der zuständigen Abteilung zugeordnet. Dies hat den Vorteil, daß vorangehende Aufträge besser „in Erinnerung“ bleiben, d.h. die Partnerschaft zwischen einem Auftragsbetreuer und einem Kundenbetreuer ermöglicht eine effizientere Zusammenarbeit, welche insbesondere dem Kunden zu gute kommt. Dies erscheint in der Realität für ein Versandhaus nicht durchführbar, da eine sehr große Zahl an Mitarbeitern benötigt würde. Dennoch wäre es wünschenswert, wenn man

genügend Personal zur Verfügung hat. Daher soll dieses Szenario hier als ideales Unternehmensmodell angesehen werden.

Nach der Anmeldung kann der Kunde in seiner Filiale beliebige Aufträge auf den Formularen erfassen und absenden. Dazu mußte er lediglich wissen, wen er bei der Anmeldung kontaktieren muß. Für den Kunden ist diese Prozedur leicht nachvollziehbar, was sich durch Kundenzufriedenheit auszahlt.

Kapitel 5 Das Architekturkonzept

Das im vorangegangenen Kapitel eingeführte Unternehmensmodell ist ein Anschauungsmodell. Es bietet die Möglichkeit, die Konzepte und Merkmale des Architekturkonzepts in einer Form zu vermitteln, die durch Anschaulichkeit und Natürlichkeit ein besseres Verständnis ermöglichen kann. Zum einen kann durch dieses Modell gezeigt werden, daß es zwischen Unternehmensstruktur und Architekturentwurf sehr viele Parallelen gibt. Zum anderen ist es möglich, konzeptionelle Probleme bei der Modellierung besser in den Griff zu bekommen, da man über eine konkretere Sichtweise verfügt, anhand derer man über abstrakte Lösungsansätze effektiver diskutieren kann. TimeLess stellt die konkrete Grundlage des Architekturentwurfs dar. Dennoch wird dessen Spezifikation weiterhin allgemein betrachtet, um eine Übertragbarkeit auf andere Softwaresysteme zu ermöglichen.

Wie bei einem Unternehmen, ist es auch bei Softwaresystemen dieser Klasse wichtig, zwischen statischen Strukturen während des normalen Betriebs und der Schaffung dieser Strukturen vor Betriebsbeginn zu unterscheiden. Aber auch während des Betriebs kommt es zu Strukturvarianz. Beim Unternehmen z.B. werden bei Bedarf neue Mitarbeiter eingestellt. Dies sollte jedoch beim normalen Betrieb transparent bleiben. Dadurch entstünde im Idealfall auf der Ebene der üblichen Geschäftsabläufe die Sicht, daß Mitarbeiter nicht bei Bedarf eingestellt werden, sondern bereits während der Filial– bzw. Abteilungsgründung eingestellt wurden. Es stünden zu jeder Zeit beliebig viele Mitarbeiter zur Verfügung. Diese Transparenz ist im realen Leben nur schwer zu erreichen. Dennoch ist die Trennung dieser beiden „Concerns“ (Schaffung von Strukturen und normaler Betrieb) stellt einen wichtigen Aspekt beim Architekturentwurf dar. In Analogie zum Unternehmen soll in diesem Kapitel zunächst die gewünschte „Betriebsstruktur“ aufgezeigt und erklärt werden. Im Anschluß daran werden die Komponenten eingeführt, die zur Schaffung dieser Strukturen benötigt werden.

5.1 Exemplarischer Systemaufbau

Das in Abbildung 5–1 dargestellte Aufbaubild zeigt eine Momentaufnahme, die alle Komponenten enthält, die während des „normalen Betriebs“ benötigt werden. Im wesentlichen sind hier alle Komponenten dargestellt, die auch im Unternehmensmodell vorkommen. Dennoch handelt es sich hierbei nicht mehr um flexible, anpassungsfähige menschliche Mitarbeiter,

sondern um programmierte Akteure, deren Verhalten formal beschrieben werden muß. Um näher auf die Komponenten eingehen zu können, wird einerseits immer wieder ein Bezug zur entsprechenden Unternehmenskomponente hergestellt, andererseits wird nachfolgend ein Szenario aus TimeLess beschrieben, das die Notwendigkeit der jeweiligen Komponente weiter untermauert.

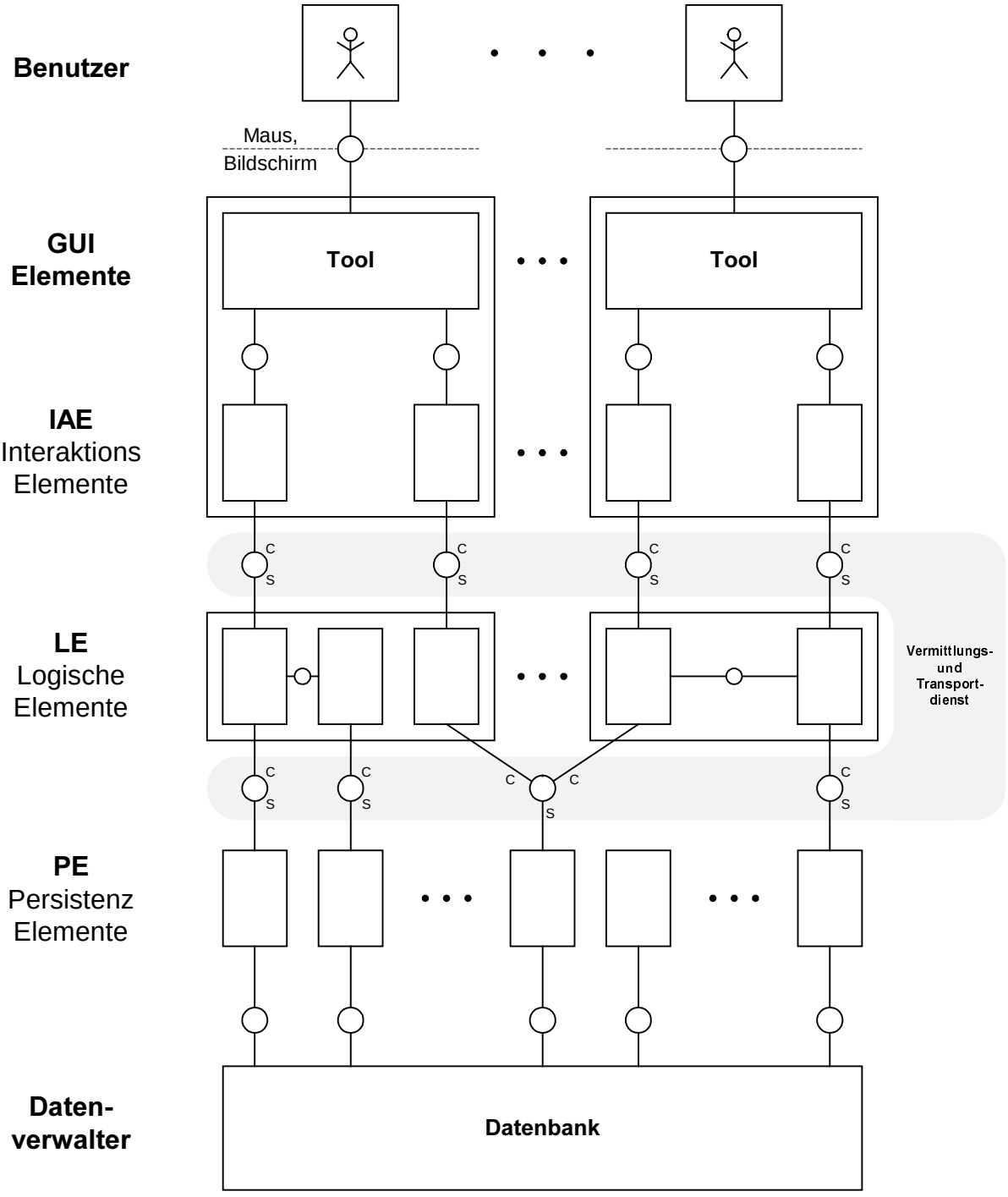

Abbildung 5–1: Exemplarischer Systemaufbau

5.1.1 Das Benutzerszenario

Im Laufe einer TimeLess–Sitzung wird der Benutzer in der Rolle des Lesers in einen Raum einer Bibliothek gehen wollen, um sich dort über bestimmte, für ihn interessante Themengebiete zu informieren. Dabei wird er mittels einer grafischen Benutzeroberfläche durch die Bibliothekswelt navigieren, bis er sich in dem entsprechenden Raum befindet. In diesem Raum wird er in Regalen Dokumente vorfinden, z.B. Bücher. Diese werden von ihm zum Teil gelesen, zum Teil möchte er aber auch nur etwas über die Bücher erfahren (zusätzliche Information). So kann es für ihn interessant sein, zu welchem Themenbereich Dokumente eines bestimmten Regals gehören.

Ebenso interessiert den Benutzer auch der Ort, an dem die Bücher stehen. Diese und andere Informationen gehören zu den impliziten Informationen, die er beiläufig während seiner Sitzung aufnimmt. So nimmt er auch implizit wahr, daß Regale eine bestimmte Anzahl Bretter haben, der Teppichboden des Raums eine bestimmte Farbe hat oder an bestimmten Dokumenten Kommentare hängen. Dieser Prozeß wird für den Benutzer nur dann nachvollziehbar sein, wenn ihm durch sein grafisches Tool die zur Orientierung notwendigen Informationen mitgeteilt werden. Die Vorgänge innerhalb des Systems, welches ihm diese Informationen aufbereitet, bleiben ihm allerdings verborgen. Welche Aufgaben die Komponenten dieses Systems bezüglich eines solchen Szenarios haben, sollen anhand Abbildung 5–1 im folgenden näher beleuchtet werden.

5.1.2 Die Komponenten

Information ist grundsätzlich klassifizierbar in Verwaltungsinformation und in verwaltete Information. In der Datenbank (im Lager) wird die gesamte Information gelagert. Um sie lagern zu können, muß eine Partitionierung der Gesamtinformation in kleinere identifizierbare (und damit unterscheidbare) Informationseinheiten erfolgen. Die Besitzer und Verwalter dieser Informationseinheiten sind die Persistenz Elemente (PEs). Diese sammeln und verwalten semantisch zusammengehörende Informationseinheiten und stellen somit für die restlichen Systemkomponenten aktive Datenverwalter semantischer Informationsgranulate dar. Die Details hierzu werden im weiteren Verlauf dieses Kapitels näher erläutert. An dieser Stelle ist es wichtig, daß man das PE als zentralen Verwalter einer bestimmten Menge von Daten erkennt.

Die von den PEs verwaltete, zustandsbeschreibende Attributmenge werden als persistente Daten bezeichnet. Damit gemeint sind die Daten, die über die Sitzungsdauer hinaus dauerhaft erhalten bleiben. In der Bibliothek wäre beispielsweise ein Buch mit sämtlichen zugehörigen Attributen durch ein PE vertreten. PEs symbolisieren Entitäten, die, wie in der Realität auch, die Eigenschaft besitzen, nur als Ganzes genutzt oder bewegt (transportiert) werden zu können. So kommen in TimeLess beispielsweise Container, wie Regale, Bretter oder Ordner aber auch Gesamtdokumente als einzigartige, identifizierbare PEs vor. Alle diese Entitäten können sowohl ihren informationellen Ort innerhalb der Anwendung als auch den materiell–energetischen Lagerort der Daten wechseln. So kann beispielsweise ein Buch von einem Regal in der Bibliothek in ein anderes verschoben werden. Dies bedingt eine informationelle Zustandsänderung des verwaltenden PEs. Das Buch–PE selbst, welches die zustandsbestimmende Attributbelegung verwaltet, könnte aber seine Daten in einer anderen Datenbank lagern, als das

Regal–PE, dem es semantisch angehört. Das PE sorgt dafür, daß die von ihm verwalteten Attribute auch physisch zusammen gelagert werden, also in der gleichen Datenbank stehen.

Der Benutzer wird sein Interesse an bestimmten Informationen dadurch formulieren, daß er ein PE als Informationsgranulat identifiziert und dessen Zustand wissen möchte. Er kann ein PE allerdings nur implizit identifizieren, da es ihm durch das Logische Element (LE) in anderer Form präsentiert wird. Die LEs vereinen die Attribute des PEs mit den aktuellen Attributen des Benutzers und den prozeßbezogenen Attributen der aktuellen Sitzung zu einer logischen Einheit. In Analogie zum Versandhaus stellen die LEs die für eine Informationseinheit zuständigen Auftragsabwickler dar. So wie die Auftragsabwickler in der Lage sind, sich bestimmte vorgangsbezogene Daten bezüglich einer Informationsanfrage für zukünftige Aufgaben zu merken, so können auch LEs zusätzliche Daten speichern. Damit besitzt das LE eine höhere Mächtigkeit gegenüber einem PE.

Beispielsweise wird ein Dokument durch ein LE in einer anwendungssemantisch angepaßteren Form repräsentiert. Dieses Dokument–LE könnte über den vom PE gelieferten Titel und Autor des Dokuments hinaus, zusätzlich noch Kommentare zum Dokument verwalten, die nur für einen bestimmten Benutzer zugänglich sind. Das LE ist somit Verwalter von Daten, welche für den Benutzer die relevanten zustandsbestimmenden Attribute eines von ihm identifizierbaren informationellen Gegenstands sind.

Ein LE vertritt höchstens ein PE. Sollte es zusätzlich Daten eines anderen PEs benötigen, so muß es mit dem für dieses andere PE zuständigen LE reden, was Abbildung 5–1 durch den Verbindungskanal zwischen LEs dargestellt ist. Dies hat den Vorteil, daß man sich auf der logischen Ebene der LEs nicht mit rudimentären PE–Diensten auseinandersetzen muß. Die semantisch höherwertigen Dienste der LEs können in vollem Umfang von anderen genutzt werden. Wie man später sehen wird, ist dies auch für die Wahrung der Konsistenz und Vermeidung von Redundanz wichtig.

Im Bild ist ebenfalls erkennbar, daß innerhalb mehrerer logischer Benutzersitzungen mehrere Exemplare von LEs vorhanden sind, welche das gleiche PE vertreten. So könnte sich ein Benutzer für den Autor eines Buchs interessieren, während ein anderer Benutzer sich für dessen Titel interessiert. Das Buch wird mit all seinen Attributen als elementares Informationsgranulat durch ein PE im System verwaltet. Bei reinem Lesezugriff muß das PE lediglich für eine Sequentialisierung[1] der Leseanfragen seitens der LEs sorgen. Schwieriger wird diese Aufgabe bei zustandsverändernden Zugriffen auf PEs. Hier müssen zusätzliche Aufgaben der PEs betrachtet werden, die zu diesem Zeitpunkt noch nicht diskutiert werden sollen.

Die logische Semantik des LEs wird durch das Interaktions Element (IAE) um die, für das Tool notwendige, grafische Semantik erweitert. Dies hat primär den Zweck, dem Tool–Entwickler eine möglichst einfache Darstellung der in der Anwendung vorkommenden Entitäten zu ermöglichen. So besitzt ein Regal–LE ein Farbattribut, welches in Form eines Textstrings verwaltet wird. Das zugehörige Regal–IAE wandelt dieses Stringattribut in einen RGB[2]–Wert

[1] Bei der Vorstellung, das PE sei ein menschlicher Verwalter, muß man davon ausgehen, daß immer nur eine Anfrage zu einem Zeitpunkt bearbeitet werden kann. Als programmierter Akteur, hängt diese Aufgabe (Sequentialisierung von Anfragen), stark vom Trägersystem sowie der eingesetzten Technologie ab.

[2] RGB: steht für die Grundfarben Rot–Grün–Blau, durch welche jede erdenkliche Farbe codiert werden kann, die durch eine entsprechende Abstufung jeder der drei Grundfarben erreicht wird.

um, welchen das Tool benutzt, um die Farbe des Regals darzustellen. Ein textbasiertes Tool könnte die Farbinformation lediglich textuell anzeigen, ein grafisches Tool aber auch in Farbe. Trotzdem wird auf logischer Ebene nur ein einziges Attribut verwaltet. Diese Aufgabe haben im Versandhaus die Kundenbetreuer übernommen, die unterem durch ein stellenweise redundantes Dienstangebot versucht haben dem Kunden eine möglichst einfache Auftragsformulierung zu ermöglichen.

Die IAEs sorgen somit für eine Übersetzung der logischen Dienste der LEs in eine Sprache und Semantik, die eine optimale Entwicklung des Tools ermöglicht. Daher müssen IAEs in der Sprachwelt des Tools zurechtkommen. Das hat die technologische Konsequenz, daß das Verhalten eines IAE in der gleichen Sprachumgebung beschrieben werden muß, wie das Tool auch. Wenn das Tool z.B. in C++ implementiert wird, dann muß das IAE auch mit C++ implementiert sein. Dies ist eigentlich keine zwingende Voraussetzung, aber dadurch bleibt die Verhaltensbeschreibung des Tools frei von unerwünschten Sprachübersetzern.

Eine vom System verwaltete Informationseinheit kann also durch verschiedene Verwaltungsebenen in drei verschiedenen Formen auftreten, als PE, LE und als IAE. Diese drei Verwaltungsinstanzen verwalten im Prinzip die gleiche Entität und stellen semantisch unterschiedliche Operationen zur Verfügung, die jeweils von dem darüber liegenden Verwaltungsakteur benutzt werden.

Bislang nicht angesprochen wurde das Verfahren, mit welchem die verschiedenen Verwaltungsinstanzen miteinander kommunizieren. Im Bild ist zwischen PE und LE sowie zwischen LE und IAE ein grau dargestellter Bereich zu erkennen, welcher den Vermittlungs– und Transportdienst symbolisiert. Aus dem Unternehmensmodell ist bereits die Notwendigkeit eines solchen Vermittlungsdienstes bekannt, da die verschiedenen Instanzen räumlich voneinander getrennt sind. In der Realität kann man sich nicht vorstellen, daß ein in seiner Filiale in Walldorf tätiger Sachbearbeiter, der Dienste seines in Frankfurt sitzenden Großzentrums in Anspruch nehmen will, persönlich nach Frankfurt fährt um eine Information in Erfahrung zu bringen. Dieser wird vielmehr einen Telefon– oder ähnlichen Dienst benutzen, um an die von ihm benötigten Information zu gelangen. Dabei sollte er sich nicht um den Verbindungsaufbau kümmern müssen. Statt dessen steht ihm ein Vermittlungsdienst zur Verfügung, der die Kommunikationsinfrastrukur bereitstellt und für eine reibungslose Kommunikation sorgt. Der im Bild dargestellte Vermittlungsdienst hat die gleiche Aufgabe innerhalb des Systems. Dabei müssen die drei Kommunikationspartner lediglich wissen, wie man den Vermittlungsdienst benutzt[1]. Dabei sollte darauf geachtet werden, daß jederzeit ein Wechsel des Vermittlungsdienstes möglich bleiben muß. In Anlehnung an die derzeit stattfindende Privatisierung des deutschen Fernmeldewesens erkennt man, daß die Notwendigkeit verschiedene Fernmeldedienste zu benutzen, nicht unbedingt vorhersehbar sein muß. Diese Trennung stellt einen weiteren zu beachtenden „Concern“ dar.

Die Kommunikation der PEs mit der Datenbank hängt ebenfalls stark von der eingesetzten Datenbanktechnologie ab. Hierfür stehen jedoch einige technologisch relevante Verfahren zur

[1] Der Vermittlungsdienst ist stark von technologischen Fragen und Fähigkeiten abhängig. Da es sich hierbei um ein sehr umfangreiches Gebiet handelt, befaßt sich eine eigens ins Leben gerufene Gruppe mit Verteilung und Vermittlung von Diensten auf Anwendungsebene: die Object Management Group (OMG). Eine Bewertung der relevanten Aspekte findet man in [BMD96].

Verfügung, deren Verwendung man sicherlich anstreben sollte, um eine möglichst hohe Unabhängigkeit vom eingesetzten Datenbanksystem zu wahren.

Die Schnittstelle zwischen Tool und Benutzer kann jede beliebige vom Tool unterstützte Mensch–Maschine–Schnittstelle sein. In der Regel wird diese aus Tastatur, Maus und Bildschirm bestehen, die eventuell durch Lautsprecher und Mikrofon ergänzt wird.

Soweit der Schnellüberblick eines exemplarischen Systemaufbaus. In den folgenden Abschnitten wird allgemein auf Zuständigkeiten der einzelnen Systemkomponenten näher eingegangen. Zu Beginn wird ein weiteres Aufbaubild gezeigt. Dieses stellt alle Systemkomponenten dar, welche während des anwendungssemantischen Betriebs benötigt werden. Hier kommen einige Komponenten hinzu. Außerdem wird das bisher eingeführte PE sowie das IAE spezialisiert.

5.2 Aufbau der Betriebskomponenten

Abbildung 5–2 zeigt die Komponenten des Systems, die während einer Benutzersitzung erforderlich sind. Dieses Aufbaubild soll als Referenzbild gesehen werden, denn spätere Erläuterungen werden sich immer wieder auf dieses Bild beziehen. Ein solches Gesamtbild läßt es zu, die beschriebenen Komponenten besser einordnen zu können und ihre Rolle im Gesamtsystem besser zu verstehen. Das Aufbaubild nutzt eine neue syntaktische Darstellung zweier Kanaltypen. Diese neue Darstellung wird beschrieben in Anhang C: „Darstellung von Kanälen“.

Die nachfolgenden Erläuterungen werden sich ebenso immer wieder auf das ER–Modell in Abbildung 5–3 beziehen. Dieses enthält bis auf die jeweiligen Verwalter die gleichen Komponenten (in Form von Entitäten), wie das Aufbaubild. Durch die Darstellung als ER–Diagramm zeigt es die genauen Beziehungen zwischen den Komponenten, die zusätzlich zum Verständnis des Architekturkonzepts beitragen können. Dieses ER–Diagramm wird nicht explizit beschrieben, sondern dient lediglich zum Verständnis des Texts. Man erkennt auch, daß das ER–Diagramm große strukturelle Ähnlichkeit mit den Versandhausstrukturen besitzt.

Das Aufbaubild in Abbildung 5–2 zeigt vier dunkelgrau dargestellte Akteure, welche die vier Hauptkomponenten des Systems darstellen. Sowohl der Tool–Akteur als auch der Interaktions–Akteur befinden sich in der Client Schicht. Der Logische Akteur befindet sich in der Logischen Schicht und der Persistenz–Akteur mit den Daten der Datenbank befindet sich in der Persistenz– und Datenverwaltungsschicht. Diese Schichten werden nachfolgend einzeln behandelt.

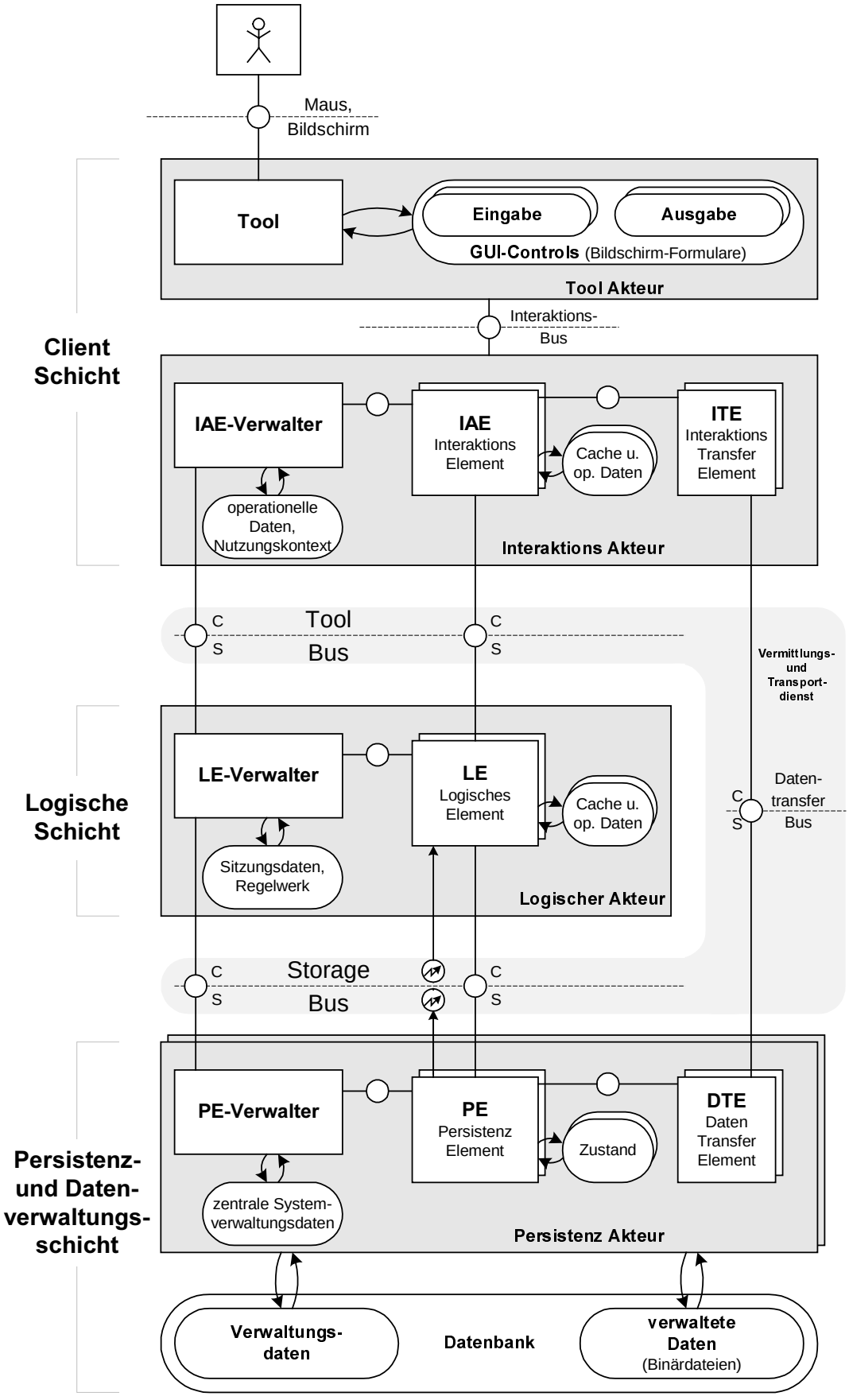

Abbildung 5–2: Die Betriebsstruktur

Allen drei Schichten gemeinsam ist ein Verwaltungsakteur analog der jeweiligen Verwaltungsabteilung im Unternehmen. Dieser ist im Bild jeweils als IAE–, LE– bzw. PE–Verwalter bezeichnet. Diese stellen, analog der Haus– und Personalverwaltung des Großzentrums im Unternehmensmodell, anwendungsübergreifende Dienste zur Verfügung, die innerhalb der jeweiligen Hauptakteurs–Komponente benötigt werden. So sind diese Verwalter in der Lage z.B. Vermittlungs– bzw. Transportdienste anzubieten, oder auch Ereignisse an die von ihnen verwalteten Elemente weiterleiten. So wie man sich eine Filialenverwaltung vorstellt, die zentral dafür sorgt, daß Postsendungen immer mit dem billigsten Kurierdienst versandt werden, so sorgt auch der IAE– bzw. LE– bzw. PE–Verwalter dafür, daß der am schnellsten verfügbare Vermittlungsdienst benutzt wird.

Man erkennt eine gewisse strukturelle Ähnlichkeit der drei Hauptakteure, wie man sie bereits bei den Elementen selbst im Schnellüberblick gesehen hat. Diese symmetrischen Strukturen sind begründet durch eine Natürlichkeit, die durch die Nachbildung der bereits im Unternehmensmodell vorgestellten real existierenden Firmenstrukturen entsteht.

5.3 Die Client Schicht

Als Anforderung an das System, wurde eine Trennung von Präsentation und Logik genannt. Diese Trennung wurde nicht nur im Rahmen der Entwickleranforderungen erwähnt, sondern auch in dem der Kundenanforderungen. Dies liegt darin begründet, daß der Kunde die Möglichkeit besitzen möchte, eigene, an seine Bedürfnisse angepaßte Benutzerschnittstellen (Tools) zu konzipieren und entwickeln. Um „Tool–Entwickler“ nicht mit schwer nachvollziehbarer, technisch orientierter Semantik zu belasten, erfolgt auf der Client–Seite die Trennung in den Tool–Akteur und den Interaktions–Akteur, welcher die Filiale des Unternehmensmodells darstellt. Der Tool–Akteur kann vom Kunden bzw. sonstigen Fremdanbietern entwickelt werden, wenn die vom Systemlieferanten angebotenen Tools den Bedürfnissen des Kunden nicht genügen. Die Tools, die dabei zur Verwendung kommen, nutzen die Dienste des Interaktions–Akteurs über die Interaktions–Bus Schnittstelle. Der Interaktions–Akteur kann allerdings selbst von Fremdanbietern realisiert werden, falls vom Systemhersteller kein geeigneter Interaktions–Akteur geliefert werden kann (z.B. weil dieser nicht in der gewünschten Technologie zur Verfügung steht). Eine solche eigene Realisierung durch Fremdanbieter setzt voraus, daß die Schnittstellen zwischen diesen Akteuren vom Systemhersteller offengelegt sind. Dies betrifft sowohl den Interaktions–Bus als auch den Tool–Bus, die im Anschluß an die Schichten in Abschnitt 5.6 näher behandelt werden.

5.3.1 Das Tool

Um dem System seine Wünsche mitteilen zu können, füllt der Benutzer, wie im Unternehmensmodell auch, Formulare aus, die in diesem Fall als GUI–Controls, bzw. GUI Elemente realisiert sind und vom Tool verwaltet und dargestellt werden. Die Tool stellt somit den Formularzugriffsakteur dar, dem der Benutzer per Maus und Tastatur seine Wünsche mitteilt. Um einen Dialog in beide Richtungen zu ermöglichen, teilt das Tool dem Benutzer Zustandsveränderungen optisch am Bildschirm sowie akustisch per Lautsprecher mit. Dieser für den Benutzer wahrnehmbare Zustand wird durch Vermittlung der Wertbelegung der GUI Elemente

auf die Sinnesorgane des Benutzers durch das Tool mittels der Mensch–Maschine Schnittstelle realisiert.

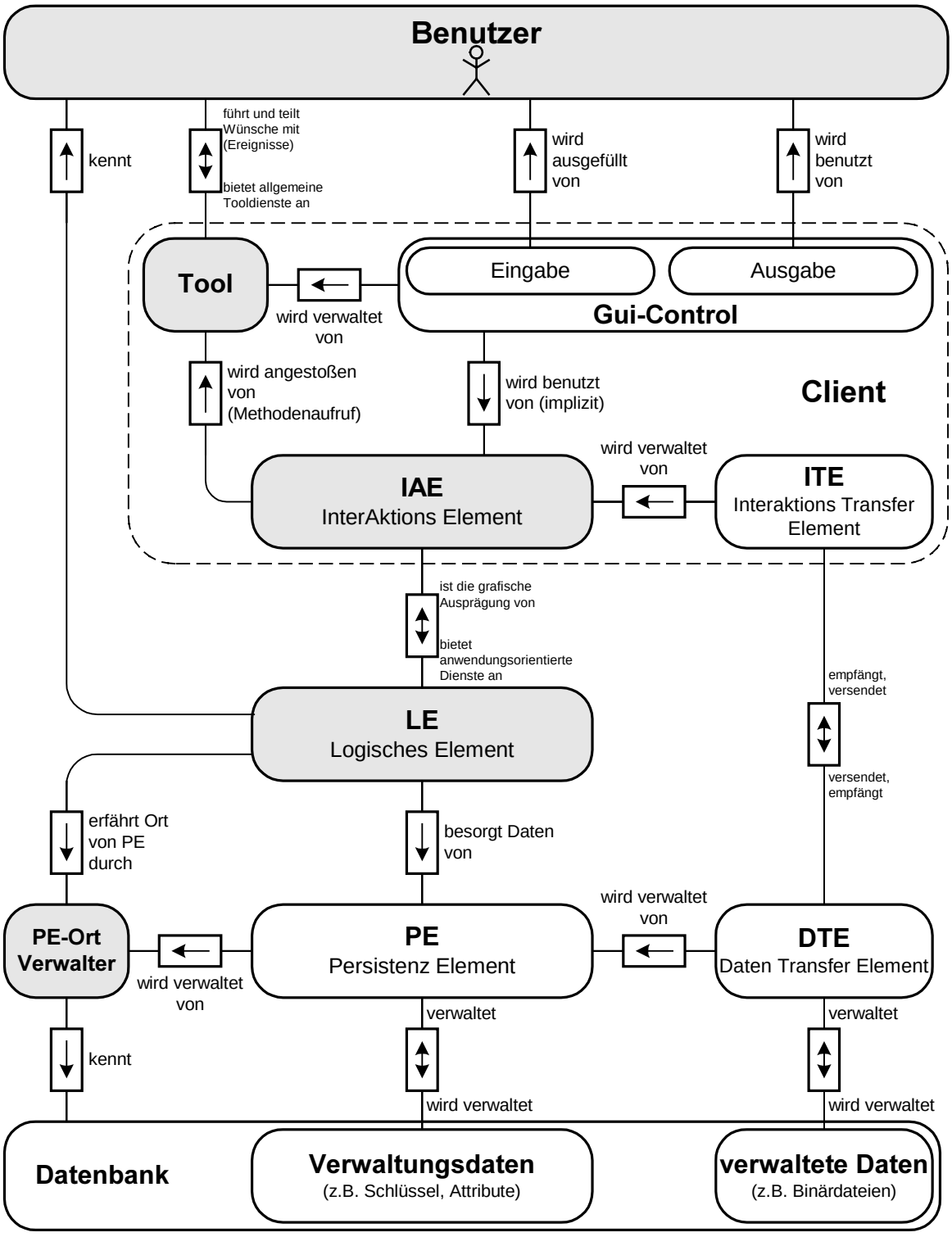

Abbildung 5–3: Beziehungen zwischen Entwurfkomponenten

Die Wertbelegung der GUI Elemente kann auf zwei verschiedene Arten geschehen, was durch die Unterscheidung zwischen GUI Eingabe– und GUI–Ausgabeelemente darstellen soll. Zum einen kann der Benutzer den Zustand von Eingabeelementen verändern. Zum anderen können Ausgabeelemente durch Abfragen der Zustandsbelegung der IAEs über den Interaktions–Bus ebenfalls durch das Tool verändert werden. Somit stellt das Tool im wesentlichen einen Vermittler zwischen Benutzer und Interaktions–Akteur dar, wobei die Vermittlung über GUI Elemente realisiert wird. Die im ER–Modell dargestellte implizite benutzt–Relation zwischen IAE und GUI–Control zeigt die logische Sicht der GUI Nutzung. Diese Relation wird explizit durch die Zugriffssemantik des Tools realisiert. Dies ist wegen der harten Trennung von Tool und Interaktions–Akteur notwendig.

5.3.2 Der Interaktions–Akteur

Der Interaktions–Akteur hat im wesentlichen zwei Aufgaben (analog den Kundenbetreuern des Versandhauses):

- Eine Erweiterung der Semantik des Logischen Akteurs um die Interaktionssemantik, die vom Tool benötigt wird. Diese Aufgabe kann prinzipiell als optimale Anpassung von logischen Diensten an die Bedürfnisse des Tools angesehen werden. Hierbei entsteht, wie im Unternehmensmodell auch, eine Redundanz von Diensten, die allerdings für das Tool einfacher zu nutzen sind.

- Die Beherrschung der Kommunikation mit dem Logischen Akteur mittels Vermittlungs– und Transportdienst. Da die Unabhängigkeit von bestimmten Vermittlungsdiensten gefordert wird, müssen Komponenten des Interaktions–Akteurs eine Austauschbarkeit ihrer Kommunikationskomponente vorsehen.

Diese Aufgaben werden verteilt auf das Interaktions Element, auf das bisher noch nicht erwähnte Interaktions–Transfer Element sowie auf den IAE–Verwalter.

5.3.3 Der IAE–Verwalter

Wie der Name bereits andeutet, handelt es sich bei diesem Akteur um den Verwalter der IAEs. Dieser ist primär für die später noch zu erläuternde Strukturvarianzverwaltung zuständig (analog der Personalabteilung im Unternehmen). Er stellt aber auch als zentraler Akteur dieser Schicht Dienste bereit, die sowohl von IAEs als auch vom Tool genutzt werden. Insbesondere sei hier der Nutzungskontext genannt. Dieser beinhaltet Information über die aktuelle Sitzung, den Erfolg und Mißerfolg von Aufträgen und sonstigen, während einer Sitzung anfallenden Daten. Hier werden auch Daten gespeichert, die für den Vermittlungsdienst von Bedeutung sind.

Der IAE–Verwalter kann sich jederzeit beim LE–Verwalter erkundigen, ob es eine Nachricht gibt, die entweder für den Benutzer, oder den IAE–Verwalter selbst interessant sein könnte. Dadurch ist auch ein Mechanismus vorhanden, der es den logischen Komponenten (Abteilung im Versandhaus) ermöglicht dem Interaktions–Akteur (der Filiale) etwas mitzuteilen. In die-

sem konkreten Fall wurde ein „Pull"–Verfahren gewählt. Dabei entscheidet sich der IAE–Verwalter über den in Abbildung 5–2 zu sehenden C/S–Kanal, wann und ob er prüft, ob für ihn eine neue Nachricht vorliegt. Es ist jedoch auch ein „Push"–Verfahren möglich bei dem der IAE–Verwalter passiv vom LE–Verwalter Nachrichtenmeldungen erhalten könnte. In diesem Fall müsste ein C/S–Kanal in die andere Richtung zusätzlich implementiert werden. Da jedoch zwischen IAE– und LE–Verwalter eine eins–zu–eins Beziehung[1] besteht, ist dies nicht zwingend erforderlich.

Außerdem erfordert ein Push–Verfahren, daß es eine Komponente in der Interaktions Schicht gibt, welche die logische Schicht als vorhanden voraussetzen kann. Analog im Versandhaus, muß die Auftragsabwicklung einen Briefkasten bei der Filiale als vorhanden voraussetzen um ihr Post senden zu können. Dies bedeutet, daß es Filialen nur mit Briefkasten geben darf. Dadurch kommt ebenso nur noch die Post für solche Sendungen in Frage. Es gibt also einige Gründe, die dagegen sprechen, daß die logische Ebene etwas in der Interaktions Ebene voraussetzen soll. Daher gilt hier der Ansatz: Die logische Ebene weiß überhaupt nichts über die Interaktions Ebene.

Daher ist es wichtig, daß Dienste der logischen Schicht grundsätzlich von jedem nutzbar sind, der in der Lage ist, das Protokoll der jeweiligen Schnittstelle zur logischen Schicht einzuhalten. Daraus folgt, daß über diese Kanäle keine Dienste angeboten werden, die keiner Prüfung auf der logischen Schicht unterliegen. Wenn man davon ausgeht, daß die Kanäle zur logischen Schicht als sichere Kanäle gelten, dann können keine Tools entwickelt werden, die durch falsches oder fehlerhaftes Verhalten das System und dessen Dienste mißbrauchen. Dies gilt sowohl für Tools mit oder ohne eigener Interaktions Schicht. Damit kann eine Zertifizierung von Tools vermieden werden.

5.3.4 Das Interaktions Element – IAE

Die IAEs sind die Vertreter von logisch faßbaren Entitäten (LEs) auf der Client Schicht. Diese ermöglichen eine Formulierung von logischen Anfragen an das System mittels grafischer Interaktionen seitens des Benutzers durch die GUI Elemente. Ein IAE kann als eine Art Übersetzer gesehen werden, der mögliche Darstellungsweisen des Tools kennt. Dadurch ist das IAE in der Lage, Tool-nahe Dienste anzubieten, die es in eine einheitliche logische Systemsprache umformt. Als Analogie könnte man sich z.B. vorstellen, daß ein deutschsprachiger Autohändler (IAE) Fahrzeuge nur an deutschsprachige Kunden (Tools) verkaufen kann. Ein anderer Händler spricht vielleicht zwei Sprachen und kann somit zwei Sprachklientele bedienen. Der Kfz–Hersteller (LE) jedoch wird sich sicherlich nicht dafür interessieren, welche Kunden welche Sprache sprechen. Übertragen auf das IAE heißt das, daß ein IAE sowohl die technische als auch die semantische Sprache des Tools kennen muß. Beispielsweise werden IAEs für ein Tool, welches den Systemzustand nur in Textform darstellt, nicht die Mächtigkeit besitzen müssen, wie beispielsweise die IAEs eines VRML[2]–basierten Tools.

[1] Diese Beziehung wird erst bei der Darstellung der Schaffungskomponenten gezeigt, kann aber bei Bedarf natürlich schon jetzt in Abbildung 5–5 nachvollzogen werden.

[2] VRML = Virtual Reality Meta Language. Diese Sprache wird sehr häufig zur Beschreibung von drei–dimensionalen Welten und deren grafische Darstellung am Bildschirm benutzt. Mit Hilfe dieser Sprache wäre es sogar möglich ein Tool zu schreiben, das eine 3D–Navigation durch eine virtuelle TimeLess–Bibliothek ermöglicht.

In TimeLess kann beispielsweise der logische Verwalter eines Ordners (Ordner–LE) die Möglichkeit anbieten, genau eines der in seinem Ordner vorkommenden Dokumente öffnen zu können. Das zugehörige Ordner–IAE kann dem Tool zusätzlich anbieten, alle Dokumente zu öffnen. Dies führt dann zwar weiterhin dazu, daß vom LE jedes Dokument einzeln geöffnet wird, doch das Tool bekommt den Eindruck es könnte alle Dokumente auf einmal öffnen.

Hier gilt also das, was bereits im Unternehmen erkannt wurde: Das LE muß ein logisch vollständiges Dienstrepertoire anbieten, das IAE hingegen hat eher das Ziel komfortable, zum teil redundante Dienste anzubieten.

Der im Versandhausmodell vorhandene direkte Zugriff der Kundenbetreuer auf die Formulare wird in diesem Konzept durch einen Mittler, nämlich dem Tool vorgenommen. Das IAE nutzt die GUI Elemente nicht direkt, sondern implizit. Dies wird durch Mitteilung des Zustands der Eingabe Controls an das IAE erreicht. Dies geschieht durch das Tool, welches darüber hinaus den Zustand der Ausgabe Controls verändert, nachdem es vorher den Zustand des IAEs erfragt hat.

IAEs müssen ebenfalls in der Lage sein, mittels Vermittlungsdienst Aufträge an das LE weiterleiten zu können. Wie in Abbildung 5–2 zu erkennen, findet die Kommunikation zwischen IAE und LE über den Tool–Bus statt. Daher muß das IAE die Semantik dieser Schnittstelle kennen und nutzen können. Dabei muß man anwendungsorientierte und technische Kommunikationssemantik unterscheiden. Die anwendungsorientierte Semantik hängt nur von der Anwendung ab und wird durch Klassifizierung der IAEs erreicht (z.B. Dokument–IAE, Ordner–IAE, Regal–IAE usw.).

Die technische Semantik hingegen hängt im wesentlichen vom Vermittlungsdienst ab. So erfordern verschiedene technische Vermittlungsdienste speziell für sie geeignete Protokolle. Eine Warenbestellung im Versandhaus kann vom Kunden direkt an den Kundenbetreuer übergeben werden. Dieser leitet die Bestellung dann mit einem speziellen Dienst weiter. Bei einem Telefondienst wird er den Auftrag in einem anderen Format vermitteln, als bei einem Telefax– oder Kurierdienst. Wenn er alle diese Dienste nutzen möchte, so bräuchte er ein Telefon, ein Telefaxgerät als auch Umschläge für den Kurierdienst. Ebenfalls bräuchte er das Wissen wie er welches Gerät benutzt.

Die Trennung dieser beiden „Concerns“ wird durch eine Schichtung innerhalb des IAEs realisiert. Diese Schichtung sollte einen Austausch der Kommunikationskomponente erlauben, um die Nutzung anderer, evtl. vom Kunden vorgegebener Vermittlungsdienste zu ermöglichen.

Eine weitere Aufgabe der IAEs stellt das Caching dar. Dabei handelt es sich um die Zwischenspeicherung bzw. Pufferung von Information. Sowohl Daten, die bereits vom Tool angefordert wurden, als auch Daten, die „vielleicht“ vom Tool noch verlangt werden, können in diesem IAE–Cache zwischengespeichert werden[1]. Der Algorithmus dieses Cache–Mechanismus muß im Einzelfall an die anwendungssemantischen Gegebenheiten angepaßt werden. Der Einsatz von Caching–Mechanismen wird nur bei Performance–Problemen benötigt – z.B.

[1] Ein möglicher Ansatz um beispielsweise dem Tool nur Dienste anzubieten, die der Benutzer im aktuellen Kontext auch ausführen kann, wird durch Caching des Nutzungskontexts im IAE erreicht. Dieser Ansatz ist in Anhang B näher erläutert.

bedingt durch einen sehr langsamen Vermittlungsdienst (Netzwerk) – und stellt somit keine vorrangige Aufgabe der IAEs dar.

5.3.5 Das Interaktions–Transfer Element – ITE

Das Interaktions–Transfer Element, kurz ITE, ist eigentlich eine Spezialform von IAEs. Dieses ist für den Transfer großer unstrukturierter Datenmengen gedacht. Dieser ist der Empfänger beispielsweise von Dokumenteninhalten. Im Versandhaus waren dies die Waren, die im Lager abgelegt wurden. Das ITE bietet keine Schnittstelle für die Nutzung durch das Tool an. Diese wird vielmehr innerhalb des zuständigen IAEs angeboten.

Ein Dokument–IAE könnte beispielsweise die Möglichkeit anbieten, den Dokumenteninhalt an ein vom Tool zu nennenden Ort abzulegen (z.B. auf die Festplatte des Benutzers). Daraufhin nutzt es ein ITE, welches den Transport dieses Dokumenteninhalts koordiniert.

An diesem Beispiel erkennt man den Grund für die Notwendigkeit des ITEs: die Koordination des Transports großer Datenmengen. Man erkennt im Aufbaubild innerhalb des hellgrau hinterlegten Vermittlungs– und Transportdienstes drei Schnittstellen: den Tool–Bus, den Storage–Bus sowie den Datentransfer–Bus. Letzterer entscheidet sich von den ersten beiden, da er Transportcharakter besitzt. Der Tool–Bus als auch der Storage–Bus hat eher den Charakter eines Vermittlungsdienstes. Durch diese Unterscheidung ist es wichtig diese unterschiedlich geartete Kommunikationssemantik von unterschiedlich gearteten Kommunikationspartnern koordinieren zu lassen. An dieser Stelle erfolgt erneut eine Trennung von „Concerns“.

Das ITE kommuniziert mit einem von der Persistenz Schicht speziell für diesen Datentransfer zur Verfügung gestellten DTE. Sowohl ITEs als auch DTEs haben die Eigenschaft nur für die Dauer eines Transferprozesses zu existieren. Diese werden nur bei Bedarf erzeugt, und sofort nach erfolgreichem Datentransport wieder vernichtet.

5.4 Die Logische Schicht

Die Logische Schicht entspricht der Ebene der Auftragsabwicklungszentren im Versandhaus. Die Aufteilung in Großzentren wird bei der Betrachtung der Schaffungskomponenten näher erläutert. Im folgenden soll davon ausgegangen werden, daß der Logische Akteur (die Auftragsabwicklungsabteilung) bereits für die Annahme von Aufträgen bereit steht.

5.4.1 Der Logische Akteur

Jedem Interaktions–Akteur steht genau ein Logischer Akteur zur Verfügung. In die andere Richtung gilt das gleiche: jeder Logische Akteur „bedient“ die Anfragen genau eines Interaktions–Akteurs. Durch diese eins–zu–eins Beziehung wird ermöglicht, daß der Logische Akteur für die Sitzung genau eines Benutzers zuständig sein kann. Dies ist ein sehr wichtiger Faktor, da die LEs, die Bestandteil des Logischen Akteurs sind, wissen müssen, für welchen Benutzer sie Verwalter sind. Eines der Hauptkonzepte bei diesem Architekturkonzept ist es, einem Benutzer nur diejenigen Zugriffe zu ermöglichen, für die er eine Berechtigung besitzt

bzw. Zugriffe, die das System im aktuellen Kontext durchführen kann. Dieser Kontext wird benötigt, um eine Prüfung der Durchführbarkeit des Auftrags zu ermöglichen. Die Komponenten des Logischen Akteurs bieten nur solche Dienste an, die sie nach erfolgreicher Prüfung als zulässig erkannt haben.

Der Kontext, bzw. das Umfelde eines Auftrags wird durch mehrere Faktoren bestimmt:

- Welcher Gegenstand ist im Rahmen des Auftrags gemeint, bzw. über welchen Gegenstand sollen Informationen ermittelt werden? Der Benutzer identifiziert passive Gegenstände und verlangt über diese bestimmte Informationen. In jedem Auftrag muß ein Gegenstand identifiziert werden.

- Wer hat den Auftrag abgesetzt? Dabei ist nicht die Quelle des Auftrags gemeint, (wie z.B. ein IAE) sondern vielmehr der eigentliche Auftraggeber: der Benutzer. An diesen werden Berechtigungen vergeben, die vor der Auftragsbearbeitung geprüft werden müssen. Sie können verhindern, daß bestimmte Aufträge durchgeführt werden können.

- Zu welcher Berechtigungsklasse gehört der Auftrag? Beispielsweise könnte es verschiedene Aufträge geben, die eine Leseberechtigung erfordern.

- Wie lauten die konkreten Auftragsparameter? Hängt die Zulässigkeit des Auftrags von weiteren durch den Auftrag vermittelten Parametern ab, so müssen diese als Prädikate einer Berechtigungsprüfung verfügbar sein.

Aufgabe dieser Diplomarbeit ist nicht, auf konkrete Realisierungsmöglichkeiten eines Berechtigungssystems einzugehen. Dennoch ist es beim Entwurf der Architektur vorzusehen, daß ein Berechtigungswesen essentieller Bestandteil eines auf dieser Architektur basierenden Produkts sein wird. Der Logische Akteur ist in diesem Entwurf die Stelle, die alle Kontextdaten für ein Berechtigungswesen zur Verfügung stellen kann. Da der Logische Akteur nur für *einen* Benutzer zuständig ist, kann er bereits vor einer Auftragserteilung durch den Interaktions–Akteur mitteilen, ob ein solcher Auftrag durchführbar ist. Das heißt, daß der Interaktions–Akteur vom Logischen Akteur die Menge der durchführbaren Dienste erfragen kann, bevor sie überhaupt vom Benutzer in Auftrag gegeben werden. Somit ist eine dynamische Bekanntgabe des aktuellen Dienstrepertoires möglich. Dies versetzt die Komponenten der Client Schicht in die Lage, dem Benutzer am Bildschirm nur die Dienste anzubieten, die er ausführen darf, bzw. die aktuell vom System ausgeführt werden können.

Viele andere Systeme gehen einen andern Weg – den Weg der optimistischen Auftragserteilung. Das bedeutet, man läßt den Benutzer Aufträge absetzen, ohne vorher zu wissen ob sie durchführbar sind. An verschiedenen Stellen im System werden Prüfungen durchgeführt, die dazu führen können, daß der Auftrag mißlingt bzw. nicht durchgeführt werden kann. Dies wird dem Benutzer dann oft mit sehr störenden Hinweisen mitgeteilt, die meistens vom Inhalt her nichts mit der Auftragssemantik zu tun haben und darüber hinaus erst mit großer Verzögerung beim Benutzer erscheinen. Dies führt zu vermeidbaren Frustrationen seitens des Benut-

zers. Für viele Benutzerschnittstellen ist dieser optimistische Weg dennoch ausreichend. Das hier vorgestellte Konzept ermöglicht sowohl eine pessimistische als auch eine optimistische Auftragserteilung.

Zum besseren Verständnis soll ein Beispiel dienen. Ein Benutzer ein Dokument lesen und drückt den entsprechenden Knopf auf seinem Bildschirm. Einige Zeit danach erhält er den Hinweis, daß er keine Berechtigung besitzt dieses Dokument zu lesen. Im hier vorgestellten Entwurf, bietet der Logische Akteur die Möglichkeit, bereits im Vorfeld durch das Tool über den Interaktions–Akteur zu erfahren, ob das Dokument gelesen werden darf. Wenn nicht, kann das entsprechende Bildschirmelement (z.B. ein Button) deaktiviert werden. Auf diese Weise kann dem Benutzer bereits im Vorfeld kenntlich gemacht werden, welche Operationen er durchführen kann und welche nicht. Er sieht am Bildschirm nicht das potentiell verfügbare vollständige Dienstrepertoire, sondern nur die Teilmenge des von ihm aktuell ausführbaren Repertoires.

5.4.2 Der LE–Verwalter

Als zentraler Verwalter innerhalb des Logischen Akteurs, werden vom LE–Verwalter nicht nur die LEs verwaltet, sondern auch die Sitzungsdaten und das Regelwerk. Zu den Sitzungsdaten gehören die Informationen, die zum soeben charakterisierten Berechtigungsskontext gehören. Hier ist beispielsweise abgelegt, welcher Benutzer der aktuelle Sitzungsinhaber ist.

Das Regelwerk in einem Unternehmen wird benötigt, um einen Mechanismus an der Hand zu haben, der es ermöglicht nach bestimmten Richtlinien vorzugehen. Wenn man als Beispiel das TimeLess Land eines Unternehmens nimmt, so kann es vom Unternehmen vorgegeben werden, daß alle Gebäude maximal 4 Stockwerke haben, oder in einem Saal mindestens 2 Regale stehen müssen. Zur Einhaltung dieser Vorgaben, müssen diese in einem formalen Regelwerk festgelegt werden. Dieses kann der LE–Verwalter bei Bedarf jedem zur Verfügung stellen, der es als Grundlage seiner Prüfungen benötigt. Dies können sowohl die von ihm verwalteten LEs sein, aber auch der IAE–Verwalter, um dem Benutzer Regelwerksdaten mitteilen zu können.

Der LE–Verwalter ist insbesondere für LEs die zentrale Anlaufstelle zur Anforderung weiterer Informationen. So erfährt ein LE beispielsweise auch für welchen Benutzer es zuständig ist. Dies ist in Abbildung 5–3 als „kennt“–Beziehung zwischen LE und Benutzer dargestellt. Dies ist somit eine implizite Relation, denn eigentlich kennt nur der LE–Verwalter den Benutzer.

Im ER–Diagramm ist zu erkennen, daß ein LE den Ort eines PEs, von dem es Daten besorgen möchte, vom PE–Ort–Verwalter erfährt. Auch dies erfährt ein LE implizit durch einen Dienst des LE–Verwalters. Dieser reicht Aufträge der LEs an den PE–Ort–Verwalter weiter und liefert das Ergebnis an das LE zurück. Dies geht sogar so weit, daß ein LE nicht direkt Daten vom PE besorgt, sondern den Auftrag, der an das PE gesendet werden soll geht zuerst an den LE–Verwalter. Dieser verpackt den Auftrag entsprechend den Erfordernissen des Vermittlungsdienstes. Das Ergebnis leitet er wieder an das LE zurück.

Durch diese zentralen Dienste des LE–Verwalters ist zu erkennen, daß hier wieder eine Trennung von „Concerns“ stattgefunden hat. Denn ein LE muß lediglich seinen Verwalter kennen sowie die Existenz des PEs, von dem es Daten besorgt (es muß das PE identifizieren können). Beide Informationen haben nichts mit Kommunikation „außer Haus“ zu tun, d.h. explizit redet ein LE (Auftragsbearbeiter) nur mit seinem Verwalter. Wie ein Sachbearbeiter in einem Unternehmen wird das LE somit von lästigen Aufgaben entbunden, wie z.B. dem Verbindungsaufbau mit dem Lager. Diese Vermittlungsdienste werden ausschließlich vom LE–Verwalter übernommen.

5.4.3 Das Logische Element – LE

LEs werden, aus den soeben besprochenen Gründen, in die Lage versetzt, ausschließlich für anwendungssemantische Aufgaben zuständig zu sein. Dadurch wird erneut dem Gedanken der „Separation of Concerns“ gefolgt. Die Verhaltensbeschreibung der LEs wird auf das eigentlich interessierende reduziert. So muß sich ein Anwendungsentwickler, dem die Spezifikation und Implementierung der LEs obliegt, nur um die Aspekte kümmern, die im Bereich der Logik seiner Anwendung liegen. Dieser Aspekt ist für eine rationelle und effiziente Anwendungsentwicklung von großer Bedeutung.

Wie bereits dargestellt, kann man das LE als einen aktiven Verwalter betrachten. Das LE erledigt Aufgaben, die der Verwaltungstätigkeit von Sachbearbeitern eines Unternehmens ähneln. Statt Auftragsdokumente und sonstige Akten zu verwalten, verwalten LEs Daten. Diese Daten sind allerdings so strukturiert, daß man sie als Attribute anwendungsorientierter, für den Endbenutzer greifbarer Entitäten bezeichnen kann. So wird es in einem TimeLess–System beispielsweise für die Entitäten Container oder Dokument ein entsprechendes LE geben, wie z.B. ein Regal–LE, ein Ordner–LE, ein Dokumenten–LE usw. Diese LEs verwalten Attribute dieser Entitäten, wie z.B. Autor, Titel und Inhalt für ein Dokument.

Das Verhalten dieser LE–Exemplare wird durch die zugehörige LE–Klasse festgelegt. Das Exemplar eines Regal–LEs gehört somit zur Klasse Regal–LE. Alle Exemplare dieser Klasse zeigen das gleiche Verhalten, das sich in den Diensten äußert, die man von einem Exemplar erwartet. So verwaltet das Exemplar eines Regal–LEs ein im TimeLess–System vorkommendes Regal und stellt Dienste zur Verfügung, die Information bezüglich des verwalteten Regals liefern. Solche Dienste ermöglichen es z.B., die Bezeichnung, die Farbe oder den Inhalt des Regals zu erfragen. Ein Dokument–LE könnte Titel, Autor und Inhalt des von ihm verwalteten Dokuments zur Verfügung stellen.

Die Dienste, die von LEs zur Verfügung gestellt werden, können in zwei Bereichen unterschieden werden: Dienste, die nach außen, also über den Tool–Bus angeboten werden, und Dienste, die nur innerhalb des Logischen Akteurs zur Verfügung gestellt werden. Diese Unterscheidung ist wichtig, da Dienste innerhalb des Logischen Akteurs keiner juristischen Berechtigungsprüfung unterliegen. Dienste innerhalb des Logischen Akteurs werden von anderen LEs als auch vom LE–Verwalter genutzt. Dienste außerhalb des Logischen Akteurs, also solche, die über den Tool–Bus zur Verfügung gestellt werden, werden von dem entsprechenden Partner–IAE des Interaktions–Akteurs genutzt.

Die Dienste, die von LEs über den Tool–Bus zur Verfügung gestellt werden, sind durch eine Schnittstellenbeschreibung der jeweiligen LE–Klasse zu spezifizieren. In dieser Spezifikation wird das Dienstrepertoire mit entsprechenden Signaturen festgelegt. Durch die Trennung des Tool–Bus in Anwendungssemantik und Vermittlungssemantik enthält diese Schnittstellenspezifikation keine technologisch bedingten Merkmale, da eine Wechsel des Vermittlungsdienstes jederzeit gewährleistet bleiben soll. Das LE kann davon ausgehen, daß die für es bestimmten Aufträge von dem Vermittlungsdienst direkt angeliefert wird. Das bedeutet, daß ein Auftragsdispatcher oder ähnliche Auftragsverteilmechanismen kein Teil des Logischen Akteurs und somit kein Teil der LEs sind. Diese Dienste muß der Vermittlungsdienst zur Verfügung stellen.

Es wurde bereits bei der Erläuterung des Logischen Akteurs die juristische Berechtigungsprüfung angesprochen. Sie ist zusammen mit der operationellen Prüfung eine der wesentlichen Aufgaben eines LEs. Das LE kann alle für die Berechtigungsprüfung notwendige Kontextinformation ermitteln. Der Kontext kann als Teil des LE–Zustands bereits vorliegen oder auch durch den LE–Verwalter ermittelt werden. Jeder vom Benutzer am Bildschirm identifizierbare Gegenstand wird von einem einzigen LE verwaltet und ist diesem somit bekannt. Information über einen identifizierten Gegenstand wird vom Verwalter des Gegenstands ermittelt. Die Auftragserteilung an den entsprechenden Verwalter übernehmen die IAEs. Da jedes LE einen Gegenstand verwaltet, kann es dessen Identität bei Prüfungen leicht als Teil des Prüfungskontext zur Verfügung stellen.

Im Rahmen des Prüfungskontext wird weiterhin der Benutzer benötigt, der den Auftrag abgegeben hat. Diesen erfährt ein LE von seinem LE–Verwalter (wie bereits erwähnt wurde). Als Empfänger von Aufträgen, kann ein LE weiter auch die zugehörige Auftragsklasse. So weiß ein Dokument–LE beispielsweise, daß die Dienste „Autor ermitteln“ und „Seitenzahl ermitteln“ beide zur Klasse der Leseaufträge gehören. Diese Klassenzugehörigkeit muß nicht statisch sein, denn die Verhaltensbeschreibung kann wiederum eine dynamische Zuordnung in Auftragsklassen ermöglichen.

Der letzte Teil des Prüfungskontext besteht aus den Parametern des Auftrags. Diese sind für das LE dadurch ermittelbar, daß diese zusammen mit dem Auftrag von der Interaktions Schicht an das LE übermittelt werden.

Ein LE kennt grundsätzlich nur ein PE, von dem es die Daten erhält, die den persistenten, also sitzungsüberdauernden Zustand des jeweiligen Objekts charakterisieren. Weiterhin steht dieses LE nur für einen bestimmten Benutzer innerhalb der aktuellen Sitzung zur Verfügung. Wenn man sich das Dokument–LE als einen Verwaltungsmitarbeiter vorstellt, dann würde dieser Mitarbeiter für einen bestimmten Kunden die Kopie eines im Aktenlager liegenden Dokuments verwalten. Dies erleichtert natürlich die Arbeit des Mitarbeiters (LEs) wesentlich. Bei einer LE–Klasse äußert sich diese Tatsache in einer vereinfachten Verhaltensbeschreibung.

An dieser Stelle kann man die berechtigte Frage stellen, warum das LE eine Kopie des Dokuments verwaltet und nicht das Original. Falls mehrmals Information bezüglich dieses Dokuments verlangt wird, macht es keinen Sinn, bei jedem neuen Auftrag erneut in das Aktenlager zu gehen, um sich die Information zu beschaffen. Ein LE wäre in diesem Fall lediglich „Durchreicher“ von Information, die es vom PE besorgt hat. Aber wenn man sich vorstellt,

der Benutzer fordert die erste Seite eines 20–seitigen Dokuments an, um kurz darauf die *nächste* Seite anzufordern, erkennt man den Nutzen, den die Verwaltung einer Kopie des Dokuments bringt. Denn das LE kann nur dann die nächste Seite kennen, wenn es sich die zuletzt gelesene Seite der Kopie „gemerkt" hat. Dies kann dadurch erreicht werden, daß das LE vorgangsrelevante Daten speichert. In diesem Beispiel kann man sich das so vorstellen, daß der Sachbearbeiter (das LE) die Kopie des Dokuments auf der Seite „offen vor sich liegen läßt", auf der zuletzt gearbeitet wurde. Dadurch weiß das LE welche Seite die aktuelle, und dadurch auch welche die nachfolgende ist. Darüber hinaus muß es beim Lesen und Blättern des Dokuments nicht jedesmal zum Aktenlager gehen um sich erneut eine Kopie des benötigten Dokuments zu machen.

An diesem Beispiel wird allerdings auch klar, daß die Granularität der zu verwaltenden Entitäten eine große Rolle spielt. So kann man sich noch vorstellen, daß ein Sachbearbeiter (bzw. ein LE) eine Kopie eines 20–seitigen Dokuments macht. Eine Kopie eines ganzen Ordners oder gar eines ganzen Aktenschranks wäre jedoch reinste Papier– und Ressourcenverschwendung (Speicherverschwendung). Dies gilt um so mehr, wenn nur ein Bruchteil der kopierten Information vom Benutzer benötigt wird. Daher müssen Datengranulate gefunden werden, die durch die Anwendung bestimmt sind, nicht durch die Datenhaltung. Dies spielt bei den zentralen Datenverwaltern, nämlich den Persistenz Elementen (PEs) eine große Rolle und wird bei der Behandlung der PEs noch einmal erläutert.

Ein letztes Aufgabengebiet der LEs wird im Aufbaubild in Abbildung 5–2 durch den Nachrichtenkanal mit dem Blitz symbolisiert. LEs haben die Möglichkeit, Nachrichten von ihren zugehörigen Partner–PEs zu empfangen. Der Empfang dieser Nachrichten erfolgt jedoch nicht über das übliche „Pull"–Verfahren, also über eine Auftrags–/Rückmeldeschnittstelle, da dies ein aktives Warten des LEs erfordern würde. Dies ist jedoch nicht erwünscht, da die Frequenz dieser Nachrichtensendungen unvorhersehbar ist. Die PEs wiederum haben einen passiven Charakter und sollen daher nicht in die Lage versetzt werden über eine Auftrags–/Rückmeldeschnittstelle ihrerseits mit ihren Partner–LEs zu kommunizieren. Somit bleibt nur das sogenannte „Push"–Verfahren eines Ereignismeldekanals. Ein solcher ist der mit den Blitzen symbolisierte im Aufbaubild. Dabei ist das PE in der Lage bestimmte Nachrichten zu senden, ohne zu wissen ob oder von wem diese Nachrichten empfangen werden. Das LE wiederum kann sich für diese Nachrichten anmelden[1] und wird bei jeder neuer Nachricht vom Vermittlungsdienst benachrichtigt. Dieser Mechanismus läßt sich in etwa mit einem Radiosender vergleichen. Auch dieser sendet bestimmte Nachrichten, ohne zu wissen ob jemand zuhört. Der Zuhörer wiederum muß lediglich eine bestimmte Frequenz auf seinem Empfangsgerät einstellen, um Nachrichten des Senders zu empfangen.

Dieses Verfahren ist sehr gut geeignet, um LEs über bestimmte Ereignisse zu informieren. Das Aufbaubild deutet zwar an dieser Stelle an, daß ein LE mit nur einem PE kommuniziert. Das ER–Diagramm hingegen zeigt, daß es sehr viele LEs geben kann, die Daten von ihrem Partner–PE besorgen. Dies ist dadurch begründet, daß es irgendwann sehr viele Benutzersitzungen geben wird, die alle ein LE besitzen, welches Daten von ein und demselben PE be-

[1] Dieses Verfahren wird meistens als Publish/Subscribe–Verfahren bezeichnet, d.h. verfassen/abonnieren. Nachrichten werden vom Erzeuger verfaßt und an alle, die Nachrichten von diesem Erzeuger abonniert haben, weitergeleitet.

sorgt. Dadurch wird es vorkommen, daß einem PE sehr viele LEs zugeordnet sein werden und es wichtig sein wird einen solchen Nachrichtenverteilmechanismus zu besitzen.

Man stelle sich z.B. vor, ein Benutzer möchte erfahren, welche anderen Benutzer derzeit das gleiche Dokument benutzen, wie er auch. Dieser Dienst kann von einem LE angeboten werden. Dieses muß jedoch sein entsprechendes Partner–PE befragen. Das PE wiederum kann die Anzahl Benutzer nur ermitteln, in dem es alle LEs, die dieses PE benutzen, ermittelt. Es sendet eine Nachricht, mit der Bitte, daß alle LEs, die für diese Nachricht angemeldet sind, sich melden. Wenn man von einer endlichen Antwortzeit ausgehen kann, dann werden dem PE bis zum Ablauf dieser Zeit alle LEs die Benutzer zurückgemeldet haben. Diese Liste kann das PE nun dem ursprünglich beauftragenden LE zurückliefern.

Da bisher die Änderbarkeit vom PE–Zustand noch nicht besprochen wurde, sei hier nur kurz der Hinweis erlaubt, daß dieser Nachrichtenkanal natürlich hauptsächlich dazu dient, die LEs zu benachrichtigen, sobald sich der Zustand eines PEs verändert hat. Daher werden über diesen Kanal vorrangig Zustandsänderungen mitgeteilt.

5.5 Die Persistenz– und Datenverwaltungsschicht

Die Persistenz– und Datenverwaltungsschicht entspricht dem Lagerverwalter mitsamt dem von ihm verwalteten Lager aus dem Informationsversandhaus. In Analogie an das Lager im Versandhaus, müssen auch hier alle Lagerbestände eindeutig identifizierbar sein. Ebenso muß auch hier unterschieden werden zwischen Information und Waren, also in diesem Fall zwischen Verwaltungsdaten und verwalteten Daten (wie Binärdateien). Im Aufbaubild ist diese Trennung durch Partitionierung der Datenbank zu erkennen. Der Begriff „Datenbank“ soll hier keine technische Bedeutung besitzen, sondern lediglich den materiell–energetischen Aufbewahrungsort der Daten symbolisieren. Nachfolgend werden alle Komponenten dieser Schicht charakterisiert.

5.5.1 Der Persistenz–Akteur

Ein Persistenz–Akteur entspricht einem Lager des Versandhauses. Im Versandhaus wird es als notwendig beschrieben, daß Lagerbestände in verschiedenen Lagern verteilt sein können. Ein großes Versandhaus ist normalerweise gar nicht in der Lage all seine Waren an einem zentralen Ort zu lagern. Auch bei großen Informationsverwaltungssystem besteht ein Bedarf an einer Verteilung der Daten. Dies wurde bereits in Abschnitt 3.2.2 als geforderte Eigenschaft genannt. Die Möglichkeit mehrere „Datenlager“ zu besitzen, wird durch eine Mehrzahl von Persistenz–Akteuren im Aufbaubild dargestellt. Ein Persistenz–Akteur besitzt jeweils einen Lagerverwalter (PE–Verwalter), sowie Lagerarbeiter (PEs), die für die gelagerten Gegenstände (Daten) zuständig sind. Die im Aufbaubild dargestellten Datentransfer Elemente (DTE) stellen eine Spezialform der PEs dar und werden in Abschnitt 5.5.4 näher behandelt. An dieser Stelle kann man sich die DTEs als Aushilfen der Lagerarbeiter (PEs) vorstellen.

Für die folgenden Betrachtungen soll davon ausgegangen werden, daß der Persistenz–Akteur bereits geschaffen ist und bereit ist Aufträge entgegenzunehmen. Der Schaffungsprozeß sowie

der Informationsdienst, der die Koordination verschiedener Persistenz–Akteure übernimmt, wird in Zusammenhang mit der Schaffung von Komponenten erläutert (Abschnitt 5.7.2).

In Analogie zu kleineren Versandhäusern, ist es auch in diesem Architekturkonzept möglich, daß nur ein zentrales Lager benötigt wird. In diesem Fall kommt nur ein Persistenz–Akteur im gesamten System vor. Es ist jedoch jederzeit möglich die Anzahl der Persistenz–Akteure bei Bedarf zu erhöhen. Dabei gilt das gleiche, wie in einem realen Versandhaus auch, die verschiedenen Teillager, also Persistenz–Akteure sind physisch verteilt.

Die Analogien zum Versandhaus sollen jedoch an dieser Stelle noch nicht beendet sein. Ein reales Versandhauslager wird sehr viel mehr Waren lagern, als es Mitarbeiter zur Verfügung hat. Das heißt, ein Mitarbeiter ist zu einem Zeitpunkt immer mit einem bestimmten gelagerten Gegenstand beschäftigt. Auch hier beim Persistenz–Akteur gilt, daß die Anzahl der Lagerarbeiter, also PEs beschränkt ist. Insbesondere bei informationellen Systemen ist leicht einzusehen, daß die Menge gelagerter Daten schnell Größen erreicht, die mit herkömmlichen Arbeitsspeichertechnologien gar nicht zu bewältigen sind. Weiterhin wird bei informationellen Systemen zu einem bestimmten Zeitpunkt nur ein Bruchteil der Daten im direkten Zugriff benötigt. Auch ein Mensch hat nie das gesamte, in seinem Gehirn gespeicherte Wissen im direkten Zugriff. Er muß den Teil, den er aktuell im direkten Zugriff benötigt, aus dem Langzeitgedächtnis ins Kurzzeitgedächtnis „kopieren“, um damit arbeiten zu können[1].

Aus diesem Grund müssen innerhalb des Persistenz–Akteurs dafür gesorgt werden, daß die Anzahl PEs beschränkt ist und dadurch eine Strukturvarianz– und Ressourcenverwaltung vorhanden sein muß. Der zentrale Akteur innerhalb des Persistenz–Akteurs, der diese Aufgabe übernimmt, ist der PE–Verwalter.

5.5.2 Der PE–Verwalter

Dem PE–Verwalter obliegt also nicht nur die Aufgabe, die PEs zu verwalten (wie sein Name andeutet), sondern auch zentrale organisatorische Aufgaben zu übernehmen. Der PE–Verwalter ist der zentrale Akteur des Persistenz–Akteurs. Er übernimmt die Aufgaben, die im Versandhaus der Lagerverwalter übernimmt. Insbesondere kennt der PE–Verwalter die ihm zur Verfügung stehenden Ressourcen zur Lagerung der Daten. Damit gemeint ist die Datenbank, die von den PEs genutzt wird. Um eine Austauschbarkeit der Datenbank jederzeit zu ermöglichen, sollten technische Eigenheiten einer bestimmten Datenbank nicht den PEs bekannt sein müssen. Daher wird auch hier, wie bereits beim LE–Verwalter, eine Trennung zweier „Concerns“ gemacht.

Ein „Concern“ ist das Kapseln von Datenzugriffen auf die PEs durch die Logische Schicht. Hierbei muß auf Aspekte, wie Wahrung von Datenintegrität, geachtet werden. Dies wird bei der Betrachtung der PEs im folgenden Abschnitt näher behandelt.

Der zweite „Concern“ ist der Datenbankzugriff, der den PEs durch den PE–Verwalter zur Verfügung gestellt wird. Damit ist die Datenbankzugriffskomponente des PE–Verwalters jederzeit austauschbar, ohne das Verhalten der PEs ändern zu müssen. In der Regel wird diese

[1] Diese Aussage erfolgt ohne wissenschaftlichen medizinischen Nachweis. Sie entspricht allerdings der subjektiven Wahrnehmung mehrerer befragten Personen.

Komponente für die verschiedenen Datenbanktypen am Markt erworben werden können. Diese Produkte müssen lediglich um die für die PEs benötigte Schnittstelle erweitert werden.

5.5.3 Das Persistenz Element – PE

Das PE läßt sich als das zentrale Element des Systems bezeichnen. Das PE ist ein Verwalter aller charakteristischen Attribute einer „gelagerten“ Entität (bzw. Gegenstand). Wenn man an TimeLess denkt, dann handelt es sich bei diesen Gegenständen unter anderem um Dokumente oder Container.

Das PE ist in der Lage, die Wertbelegungen der von ihm verwalteten Attribute auf Anfrage von LEs mitzuteilen. So ist ein Dokument–PE z.B. in der Lage, den Autoren, den Titel oder die Seitenzahl mitzuteilen.

Ein PE sorgt für die Erhaltung der Integrität seiner Attribute. Diese Integrität bezieht sich nicht auf Anwendungssemantik, sondern normalerweise nur auf technisch bedingte Eigenschaften. So kann ein vom PE verwaltetes Attribut vom Typ Integer nicht mit einem String belegt werden. Das PE ist aber ansonsten reiner „Handlanger“ der LEs, die letztlich für die Wertbelegung von PE–Attributen zuständig sind.

Das PE wurde in Analogie an das Versandhaus bereits mit dem Lagerarbeiter verglichen. Dazu muß man sagen, daß es verschiedene Arten von PEs gibt, die unterschiedliche Attribute verwalten. In anderer Form ausgedrückt gibt es verschiedene Klassen von Lagerarbeiter. Jedes PE–Exemplar gehört einer bestimmten PE–Klasse an. Alle Exemplare einer Klasse zeigen gleiches Verhalten, welches durch die Verhaltensbeschreibung der Klasse festgelegt wird.

Ein PE ist Verwalter einer eindeutig identifizierbarer Entität und ist als solches nur einmal im Gesamtsystem vorhanden. Das bedeutet z.B., daß ein Dokument–PE als Verwalter der Attribute eines bestimmten Dokuments nur einmal als solches im System aufzufinden sein wird. Sollten mehrere Persistenz–Akteure innerhalb eines Systems existieren, dann ist es wichtig, daß diese die Schaffung der PEs so koordinieren, daß eine bestimmte gelagerte Entität nur durch ein einziges PE vertreten ist.

Die Frage, die sich nun stellt, ist was man unter einer gelagerten Entität zu verstehen hat. In diesem Zusammenhang spielt der Aspekt der Granularität eine wichtige Rolle. Ein Granulat ist auf einer bestimmten Betrachtungsebene die kleinste, nicht weiter zerlegbare Einheit. Größere Einheiten können aus vielen Granularen Einheiten bestehen. Ein Granulat jedoch, ist die elementarste Einheit (manchmal wird auch die Bezeichnung „atomar“ benutzt).

Wenn man mit einem Granulat arbeiten möchte, ist man gezwungen, das gesamte Granulat für sich allein in Anspruch zu nehmen. In vielen Systemen hängen Granulate allerdings stark von zugrundeliegenden Datenstrukturen ab. Beispiele solcher Granulate sind Datensätze einer Tabelle, oder gar von der Datenbank benutzte Speicherentitäten, wie Seiten (Pages), die Daten enthalten können, die semantisch nichts miteinander zu tun haben. Dies führt bei Mehr–Be-

nutzer–Systemen immer wieder zu Problemen, da Granulate immer nur einem Benutzer zur Verfügung gestellt werden können[1].

Beispielsweise möchte ein Benutzer die Stammdaten eines Mitarbeiters verändern (z.B. Telefonnummer), während ein anderer die Arbeitszeit des gleichen Mitarbeiters eintragen möchte. Alle Daten des Mitarbeiters werden allerdings vom System als Granulat verwaltet. Dies führt zu Problemen, da Änderungen an Teilen des Granulats als Änderung des *gesamten* Granulats angesehen werden müssen. Im dargestellten Fall wäre die Verkleinerung des Granulats eine mögliche Lösung. Der Grund eine solche Granularität zu wählen, hängt meistens eng mit der Performanz solcher Systeme zusammen. Je größer das Granulat, desto geringer der Verwaltungsaufwand.

Wenn man die Größe von Granulaten verkleinert, steigt dieser Verwaltungsaufwand enorm an. So ist es möglich, daß jedes Datum ein Granulat darstellt. Für jedes Granulat müssen aber zusätzliche Daten gespeichert werden[2]. Diese erhöhen den Speicher und Verwaltungsbedarf zum Teil so stark, daß man sogar mit Leistungseinbußen rechnen muß.

Diese Ansätze der Granulatfindung sind sicherlich nicht geeignet, da sie sich an der Semantik der Datenhaltung orientieren und nicht an der Semantik der Datennutzung. Es wäre in einer Bibliothek z.B. nicht denkbar, daß ein Leser, der ein Buch lesen möchte, dies nicht kann, weil ein anderer Benutzer zum gleichen Zeitpunkt vorm gleichen Regal steht. Das Granulat stellt in diesem Fall das Regal dar. Viel eher kann man schon verstehen, daß das Exemplar eines Buchs immer nur von einem Benutzer gelesen werden kann. Das heißt, das Buch kommt als Granulat eher in Frage. Somit sind alle Bestandteile des Buchs (Hülle, Seiten, Text usw.) auch Bestandteile des Granulats. Würde man diese Situation auf die Persistenz Schicht abbilden, dann wären alle Attribute, die ein Buch beschreiben, fester Bestandteil eines Granulats. Solche Granulate werden durch PEs geschaffen. In einem informationellen Bibliothekssystem basierend auf diesem Architekturkonzept (TimeLess) wären alle Bücher in Form von PEs vertreten. Wenn man sich fragt, „Was sind PEs?“, dann kann man diese Frage meistens durch Kenntnis der Anwendungssemantik und den darin zu verwaltenden Gegenständen befriedigend beantworten.

Zur Verwaltung der Granulate sollte immer nur ein angemessener Aufwand betrieben werden müssen. Es müssen nicht mehr, aber auch nicht weniger als notwendig zusätzlicher Verwaltungsattribute gespeichert werden. Dies läßt sich an einem kleinen Beispiel verdeutlichen. Vor einem Aktenschrank liegt eine Liste aus, in der man alle Ordner eintragen muß, die man ausgeliehen hat. Es ist jedoch nicht erforderlich jede Seite innerhalb eines Ordners in der Liste zu vermerken, da man immer nur gesamte Ordner entnehmen kann. Der Aufwand den Ordner in die Liste einzutragen wird vermutlich jedem als angemessen erscheinen.

In einem zweiten Ansatz den Zugang zum Schrank ohne Liste zu realisieren, könnte man einen Schlüssel zum Schrank so ausleihen, daß man zu jedem Zeitpunkt weiß, wer den Schlüssel hat. Der Schlüsselinhaber kann über den Schrank verfügen, wie er möchte. Bevor er den Schlüssel an andere weitergibt, muß er den Schrank aber wieder in einen zulässigen Zustand

[1] An dieser Stelle sei erwähnt, daß dies mit der Sperrnotwendigkeit zusammen hängt. Da Sperren jedoch an dieser Stelle nicht näher behandelt werden sollen, wird hier nur darauf hingewiesen.
[2] Beispiele für Granulatverwaltungsdaten sind der Timestamp, der festhält, wann das Granulat zuletzt geändert wurde oder ein „benutzt“–Attribut, welches festhält, ob das Granulat aktuell in Gebrauch ist..

bringen. Nur so ist nachzuvollziehen, wer mit Ordnern dieses Schranks derzeit beschäftigt ist. Dieser Ansatz ist nur dann möglich, wenn selten Ordner von mehreren Nutzern gleichzeitig benötigt werden. Dies muß nicht unbedingt immer gegeben sein.

Ein dritter Ansatz, der ein anderes Extrem darstellt, ist die Entnahme einzelner Seiten eines Ordners. Jede entnommene Seite ist in die Liste einzutragen, damit man weiß, wer welche Seite benutzt. Dies bedeutet aber in den Fällen, in denen viele Seiten benötigt werden, daß sehr viele Einträge in die Liste gemacht werden müssen und es ebenfalls sehr viel länger dauert die Seiten den Ordnern zu entnehmen, d.h. man hält sich länger vorm Aktenschrank auf.

Die Wahl des besten Verfahrens ist tatsächlich von Fall zu Fall unterschiedlich und hängt einzig und allein von der Semantik der Nutzung ab. Dies ist der entscheidende Vorteil bei der Wahl der Granularität der PEs. Sie kann nämlich von Fall zu Fall der Anwendungssemantik angepaßt werden. So kann man sich Dokument–PEs vorstellen, die bereits elementare Einheiten darstellen. Man kann sich aber auch Regal–PEs vorstellen, die mit vielen Dokument–PEs gefüllt sind. Dadurch ist es möglich, neue Dokumente in das Regal zu stellen, während im Regal bereits vorhandene Dokumente von anderen benutzt werden können.

PEs haben während des normalen Anwendungsbetriebs rein passiven Charakter für die Logische Ebene. Das heißt sie stellen für die LEs lediglich Attributzugriffsdienste zur Verfügung und besitzen daher auch nur Speichersemantik. Wegen dieser Passivität kann zu diesem Zeitpunkt nicht von aktivem Verhalten von PEs gesprochen werden. PEs haben jedoch auf einer anderen Betrachtungsebene sehr wohl aktiven Charakter. Dieser kann allerdings erst bei der Schaffung von Komponenten bzw. bei Zustandsänderungen beleuchtet werden. Diese Aspekte werden ab Abschnitt 5.7 behandelt.

5.5.4 Das DatenTransfer Element – DTE

PEs sind in der Lage, jeden Attributtyp zu verwalten. Dennoch ist eine Spezialform von PEs erforderlich, um einen speziellen Datentyp zu verwalten. Dieser unterscheidet sich von anderen Attributen dadurch, daß er in seiner Größe nicht beschränkt ist. Dadurch ist das Anfertigen einer Kopie nicht sinnvoll und erfordert somit eine andere Art der Verwaltung. Das bereits bei den Interaktions Elementen angesprochene Datentransfer Element wird diese Aufgabe übernehmen.

Ein DTE wird vom PE erzeugt, wenn über das LE die Bereitstellung eines DTEs am Datentransfer–Bus angefordert wurde. In der Sprache des Versandhauses ausgedrückt, hat der Kunde ein Buch bestellt. Sein Kundenbetreuer (IAE) stellt ein Postfach (ITE) zur Verfügung. Dieses Postfach stellt den Lieferort dar, an welchen das Buch geliefert werden soll. Die Adresse des Postfachs teilt er seiner Auftragsbearbeitung (LE) mit, die diesen Auftrag unmittelbar an das Lager und dessen Lagerarbeiter weitergibt (PE). Dieser Lagerarbeiter besorgt sich ein Aushilfsmitarbeiter (DTE), den er beauftragt, das bestellte Buch aus dem Lager zu besorgen, es zu verpacken und mit der Lieferadresse (ITE) zu versehen und es schließlich zur Post zu bringen. Diese Aushilfe sorgt weiterhin dafür, daß das Buch auch tatsächlich beim Kunden ankommt.

Diese Aufgabe hätte das PE, bzw. der Lagerarbeiter auch erledigen können. Allerdings ist die Vorgangsabwicklung – wie in der Realität der Gang zur Post – sehr aufwendig und kostet sehr viel Zeit, in der das PE für andere Tätigkeiten nicht zur Verfügung steht. Weiterhin hat der Datentransfer–Bus andere Richtlinien als der Storage Bus, insbesondere bezüglich der Sicherheit (wird in Abschnitt 5.6 „Die Schnittstellen“ näher beleuchtet).

Ein DTE wird nur für die Dauer eines Transfers, also genau für einen Transfervorgang, benötigt. Danach wird es wieder gelöscht. Anders ausgedrückt, für jeden Transfervorgang wird eine neue Aushilfe eingestellt, die den Botengang macht. Natürlich könnte es auch eine Aushilfe sein, die bereits einen früheren Botengang gemacht hat, doch dies ist für das beauftragende PE nicht von Interesse.

5.5.5 Die Datenbank

Die Datenbank wird von den PEs benutzt als Datenlager. Einmal an diesem Ort angekommen, kann man davon ausgehen, daß die Daten auch tatsächlich persistent sind, also auch technische Störungen, wie Netzwerkfehler oder Systemabstürze überdauern. Im Aufbaubild wird dieses Lager als Speicher dargestellt, der sowohl Verwaltungsdaten als auch die verwalteten Daten beinhaltet. Dieser Speicher wird in einem realisierten System sicherlich aus einer aktiven Datenbank bestehen, die wiederum die Einspeicherung von Daten durch einen Datenzugriffsakteur erreicht. Im Kontext dieses Architekturkonzepts besitzt die Datenbank allerdings reinen speichernden Charakter und wird daher als passive Komponente angesehen.

Im allgemeinen EDV–Jargon wird mit einer Datenbank meistens ein zentrales Softwaresystem gemeint, welches im ständigen Zugriff ist auf eigens verwaltete, periphere Speichergeräte, wie Festplatten und Bandlaufwerke. Dies stellt zwar eine mögliche technische Realisierungsmöglichkeit dar, doch hier ist der Datenbankbegriff eher allgemein zu verstehen. Die Strukturierung der Daten in der Datenbank soll im Rahmen dieses Architekturkonzepts offen gelassen werden. Zwar wurde in [Langhauser97] ein für TimeLess einzusetzendes Datenbankschema entworfen, welches die Datenstrukturen beschreibt, die für eine effiziente Datenverwaltung notwendig sind. Dennoch sollen hier keine Vorschriften bezüglich des Einsatzes bestimmter Datenbanktechnologien oder dafür einzusetzende Datenschemata gemacht werden. So wäre es im Extremfall gar denkbar, daß ein flaches Dateisystem zur Lagerung der von den PEs benötigten Daten ausreichen könnte.

Der Grund, trotzdem gängige Datenbanksysteme einzusetzen liegt zum einen an deren Komplexität, zum anderen an deren Mächtigkeit. Die Komplexität eines gewachsenen Datenbanksystems macht es nahezu unmöglich ein vergleichbares System selbst zu entwickeln. Darüber hinaus wäre eine eigene Entwicklung nur unter ganz bestimmten Bedingungen auch wirtschaftlich. Insbesondere wenn man ein System mit vergleichbarer Mächtigkeit, wie die vorhandener Systeme entwickeln wollte. Daher sollen für den Einsatz einer bestimmten Datenbank maßgeblich die Leistungsfähigkeit sowie das Vorhandensein von Bedienwissen im Vordergrund stehen. Fast alle gängigen Datenbanken befriedigen den funktionellen Bedarf dieses Architekturkonzepts. Dies gilt sowohl für relationale als auch für objektorientierte Datenbanken. Dem objektorientierten Charakter der Architektur entspricht sicherlich eher eine objektorientierte Datenbank, aber die Faktoren Leistungsfähigkeit und Vorhandensein von Bedienwissen begünstigen eher den Einsatz relationaler Technologie.

Eine Strukturierung der Daten auf Datenbankebene sollte besonders dann in Betracht gezogen werden, wenn eine Suchmaschine zum Einsatz kommen soll. Eine Suchmaschine zeichnet sich dadurch aus, daß sie sehr große Datenmengen durchsuchen kann. Man könnte sich beispielsweise vorstellen, daß mittels Volltextsuche alle Dokumente gesucht werden sollen, die den Begriff „Architektur“ enthalten. Das Ergebnis wäre eine Liste aller zutreffenden Dokumente. Eine solche Suche würde über die PE–Schicht sehr lange dauern, da jedes Dokument aus dem Lager geholt werden muß (von einem PE). Dieses müßte auf das Suchkriterium untersucht werden und zurück ins Lager gestellt werden. Dieser Vorgang müßte für alle Dokumente wiederholt werden, bis alle überprüft sind. Man merkt schon an der Analogie des Versandhauses, daß dies keine Realisierungsmöglichkeit darstellen kann.

Während dieser Diplomarbeit konnten die Konzepte bezüglich einer Suchmaschine noch nicht weiter vertieft werden. Eine Alternative stellt derzeit die Suche auf Datenbankebene dar. Dabei tritt jedoch ein Synchronisationsproblem auf, da die verwaltenden PEs evtl. einen anderen Zustand kennen als der Zustand, der in der Datenbank steht. Ebenso werden gelöschte Daten bei einer solchen Suche unter Umständen noch mit berücksichtigt, da diese erst später durch eine „Garbage Collection[1]“ aus der Datenbank gelöscht werden. Aus diesen Gründen, hängt die Auswahl einer geeigneten Datenbank stark von der benötigten assoziativen Suchsemantik ab.

5.6 Die Schnittstellen

Bei großen Systemen ist das Vorhandensein von Schnittstellen entscheidend für die Erweiterbarkeit und Wartbarkeit. Sowohl Schnittstellen zum Anschluß der Außenwelt an das Basissystem als auch interne Schnittstellen spielen eine wichtige Rolle, wenn es darum geht, Änderungswünsche umzusetzen. Der Bedarf an langlebigen Schnittstellen wird beim Systementwurf oft nicht berücksichtigt.

Eingangs wurde bereits erwähnt, daß die Güte eines Architekturentwurfs vorrangig durch die Langlebigkeit der in ihr spezifizierten Schnittstellen charakterisiert werden kann. Im Aufbaubild in Abbildung 5–2 sind mehrere Schnittstellen zu erkennen, deren Langlebigkeit oberstes Kriterium bei den Vorüberlegungen dargestellt hat. Hierzu zählen insbesondere der Tool–Bus, der Storage–Bus als auch der Datentransfer–Bus[2]. Diese drei Bus–Schnittstellen besitzen jeweils zwei Aufgaben, die als zu trennende „Concerns“ angesehen werden müssen:

[1] Garbage Collection:= Müll Sammlung. Dieser Begriff steht für einen Mechanismus, der nur Referenzen auf nicht mehr benötigte Daten löscht. Die referenzierten Daten selbst werden asynchron erst zu einem späteren Zeitpunkt tatsächlich gelöscht. Dies hat den Vorteil, daß Löschoperationen schneller vollzogen werden, und die tatsächliche Löschung zu einem Zeitpunkt stattfinden kann, zu der viele Ressourcen frei sind (z.B. nachts).

[2] Der Begriff Bus wurde hier gewählt, um anzudeuten, daß die Verwendung eines Standardprotokolls ein beliebiges Hinzufügen von Teilnehmern (Kommunikationspartnern) ermöglicht. Somit werden durch diese Bus–Schnittstellen z.B. Erweiterungen bzw. Änderungen der Anwendungssemantik möglich ohne eine Anpassung der Schnittstellenprotokolle zu benötigen. Dies ist ein entscheidendes Kriterium für deren Langlebigkeit.

- Sie müssen einen Kommunikationsaufbau über Zuständigkeitsgrenzen hinweg ermöglichen. Hier müssen wegen der räumlichen Trennung der Kommunikationspartner auch Rechnergrenzen überwunden werden können. Dies wird im wesentlichen durch den Vermittlungs– und Transportdienst erledigt.

  Anzustreben ist dabei eine Realisierung, die den Anwendungskomponenten den Eindruck vermittelt, daß Kommunikationspartner direkt miteinander kommunizieren, ohne sich um Kanalauf– und abbau, Paketierung, Verschlüsselung und sonstiger Aspekte kümmern zu müssen. Dies kann man, analog dem ISO/OSI–Schichtenmodell, durch Schichtungen des Kommunikationskanals erreichen. Dies sollte allerdings bis zur Anwendungsschicht durch den Vermittlungs– und Transportdienst realisiert sein.

  Da es sich hierbei um Softwareschnittstellen handelt, sollte auch gesagt werden, daß man sich hier eine hohe Unabhängigkeit von bestimmten Technologien wünscht. Somit sollte als Aufgabe ebenfalls die Sprachunabhängigkeit genannt werden, bzw. Plattformunabhängigkeit (Portabilität). So ist es wünschenswert, daß das Bus–Protokoll in einer Art Metasprache spezifiziert wird, die dann auf die jeweils gängigen Technologien abgebildet werden kann. Bei der Nutzung eines geeigneten Vermittlungs– und Transportdienstes ist dies ein wichtiges Kriterium.

- Die Protokolle der Bus–Schnittstellen müssen so konzipiert sein, daß jederzeit Änderungen oder Erweiterungen der bisherigen Anwendungssemantik möglich bleiben. Es muß gewährleistet sein, daß auch ältere Anwendungskomponenten weiterhin über diese Schnittstelle kommunizieren können, ohne von Änderungen oder Ergänzungen der Anwendungsschicht behindert zu werden.

Um sich ein Bild über die Semantik der drei Bus–Schnittstellen machen zu können, ist in Anhang A: „Identifizieren und Zeigen“ ein exemplarischer Auszug der technischen Realisierung des TimeLess Prototyps zu finden. Im Abschnitt A.2 des Anhangs werden einige konkrete Details angesprochen, die an dieser Stelle weit führen würden. Dennoch kann an diesem konkreten Realisierungsbeispiel besser nachvollzogen werden, in welchen Fällen an den jeweiligen Schnittstellen identifiziert bzw. referenziert wird.

5.6.1 Der Interaktions–Bus

Der Interaktions–Bus unterliegt nicht zwingend den hohen Anforderungen an die Langlebigkeit, da die Kommunikation zwischen Tool und Interaktions–Akteur eine innere Schnittstelle der Client Schicht darstellt und somit nicht die Universalität der anderen Bus–Schnittstellen benötigt. Hier ist Universalität sogar unerwünscht, da aus den bereits erwähnten Gründen Interaktions Dienste möglichst dem Bedarf des Tools angepaßt werden müssen. Der Interaktions–Akteur bietet Übersetzungsdienste für das Tool an, ohne Rechnergrenzen überwinden zu müssen. Insofern sollen beim Interaktions–Bus Änderungen der Schnittstellenspezifikation unter bestimmten Umständen zugelassen werden (z.B. unter dem Aspekt, daß das Tool von

Fremdanbietern gebaut wird und – Technologie bedingt – von diesen eine andere Interaktions–Bus Schnittstelle benötigt wird).

Sollte das Verhalten des Interaktions–Akteurs in einer objektorientierten Sprache beschrieben sein, dann wird der Interaktions–Bus durch die öffentlichen Dienste der IAE– und ITE–Klassen sowie des IAE–Verwalters spezifiziert.

5.6.2 Der Tool–Bus

Die Kommunikation mittels Tool–Bus ist vergleichbar mit einem Telefongespräch zwischen Kundenbetreuern und Auftragsabwicklern im Versandhaus. Bei jedem Gespräch muß der Empfänger benannt werden (Adressierung) um eine Kommunikation aufzubauen. Da der Tool–Bus ausschließlich aus ein Auftrags–/Rückmeldeschnittstelle in Richtung der Logischen Schicht besteht, können nur Akteure der Logischen Schicht als Empfänger in Frage kommen.

Empfänger können nur durch direktes „Anwählen“ angesprochen werden, d.h. es gibt keinen Vermittlungsdienst, der Aufträge an die Empfänger weiterleitet. Ein IAE (Auftraggeber) muß grundsätzlich seinen Kommunikationspartner (LE) bereits vor der Auftragserteilung kennen. Man kann diesen Sachverhalt auch so bezeichnen, daß es keinen Auskunftsdienst gibt, der die „Telefonnummern“ von Logischen Komponenten mitteilt. Der einzige Weg an die Telefonnummern anderer LEs zu kommen ist durch „Hörensagen“. Das bedeutet, daß LEs, die wiederum andere LEs kennen, können als Dienst deren Telefonnummer einem IAE mitteilen.

So kann man den Verwalter eines Saals (Saal–LE) fragen, ob in ihm Regale stehen, und wenn ja, wie die Telefonnummern dieser Regal–LEs lauten. Ein anderes Beispiel dieses „Hörensagens“ kennt jeder selbst: Die Telefonnummer eines guten Restaurants oder Arztes erfährt man in der Regel nicht über die Telefonauskunft, sondern von Mitmenschen, die man wiederum persönlich kennt.

Dennoch muß an dieser Stelle wiederholt werden, daß der Vermittlungsdienst so nicht arbeiten kann. Er benötigt einen Identifikator des Empfängers, um den Telefonkanal aufbauen zu können. Dies ist jedoch für die Anwendungsschicht vollkommen irrelevant.

Dieses „Hörensagen“–Konzept funktioniert nur, wenn es zu Beginn einen Identifikatorlieferanten gibt, dessen Existenz vorausgesetzt werden kann. Diese zentrale Anlaufstelle gehört zu den Schaffungskomponenten, da sie dafür zuständig ist, daß der Logische Akteur geschaffen bzw. aktiviert wird. Sie gehört zu der Klasse der „Well–Known“–Server und wird bei der Schaffung von Komponenten in Abschnitt 5.7 erneut angesprochen.

Der Tool–Bus ist ein allgemeine Schnittstelle zum Ankopplung von Peripherie (Clients), im Gegensatz zum Interaktions–Bus, der eine spezielle Schnittstelle darstellt. Dienste des Tool–Bus müssen eine vollständige Algebra darstellen, d.h. es muß möglich sein durch eine formal beschreibbare Kombination und Reihenfolge von Aufrufen über diese Schnittstelle den Empfänger in jeden vorhersehbaren Zustand überführen zu können. Das Dienstrepertoire muß vollständig sein.

Ähnlich den Kosten eines Telefongesprächs in der Realität, kostet der Aufruf eines Dienstes am Tool–Bus sowohl Netzwerkressourcen als auch Zeit. Somit muß man bei der Formulie-

rung der Dienste am Tool–Bus berücksichtigen, daß Redundanz vermieden wird und viele Aufträge durch Pakete zusammengefaßt werden. Hiermit ist keine Paketierung im Sinne einer Protokollschichtung gemeint, sondern vielmehr das Zusammenfassen vieler kleinen Dienste zu einem Großdienst. Man würde bei der Kinoauskunft auch nicht fünf mal anrufen, um zu erfahren welcher Film in den fünf verschiedenen Kinosälen läuft. Diese Anfrage würde man zu einer Gesamtanfrage zusammenfassen. Die Spezifikation von Diensten am Tool–Bus muß daher berücksichtigen, daß durch den Vermittlungsdienst ein sehr großer Aufwand (Overhead) betrieben werden muß, um die Kommunikation eines Aufrufs zu koordinieren.

Eine weitere spezielle Eigenschaft besitzt der Tool–Bus: Dienste können durch mangelnde Berechtigungen oder anderer operationeller Gründe bei Bedarf verweigert werden. Diesbezüglich sieht die Spezifikation des Tool–Bus vor, daß es drei Klassen von Diensten gibt:

- Unsichere Dienste (ohne Prüfung)

  Hiermit sind die Dienste gemeint, die weder einer Prüfung unterliegen, noch Auskunft über die Durchführbarkeit anderer Dienste geben. Login und Logout Dienste gehören beispielsweise zu dieser Klasse.

- Sichere Dienste (mit Prüfung)

  Diese Dienste unterliegen grundsätzlich einer Prüfung. Diese Prüfung ist notwendig um zu garantieren, daß Benutzer nur die Information bekommen, die sie auch bekommen dürfen. Ebenso erreicht man durch diese Prüfung, daß das System nicht in einen unzulässigen Zustand überführt wird. Bei Verweigerung eines Dienstes wegen erfolgloser Prüfung, erhält der Auftraggeber eine Rückmeldung, die ihm den Grund nennt, daß der Dienst nicht durchgeführt werden konnte.

- Erkundigungsdienste (ohne Prüfung)

  Erkundigungsdienste sind eine spezielle Form von unsicheren Diensten. Der Grund dieser Spezialisierung liegt darin, daß Erkundigungsdienste aus der Existenz der sicheren Dienste abgeleitet werden können. Das Vorhandensein eines sicheren Dienstes impliziert die Existenz eines zugehörigen Erkundigungsdienstes. Dienste dieser Klasse müssen somit nicht explizit spezifiziert werden, sondern können direkt aus der Menge der sicheren Dienste abgeleitet werden.

  Erkundigungsdienste können verhindern, daß sichere Dienste aufgerufen werden, die z.B. wegen mangelnder Berechtigung postwendend mit einer Ablehnungsmeldung antworten. Sie können auf schnellere Weise als sichere Dienste bereits im Vorfeld darüber Auskunft geben, ob sichere Dienste ausgeführt werden können oder nicht. Dies ist ähnlich einer Bank, bei der man sich auch erst erkundigen kann, ob noch genügend Geld auf dem Konto ist, bevor man Geld abhebt. Versucht man Geld abzuheben, ohne vorher zu prüfen, muß man mit einer Ablehnung rechnen. Eine solche Ablehnung ist nachvollziehbar, wenn man mit ihr rechnet (weil man z.B. weiß, daß der Kontostand sehr niedrig ist). Dennoch wäre es vielfach wünschenswert, daß bereits vor Eingabe der Menge, die man abheben möchte, bereits bekannt ist, ob eine Abhebung grundsätzlich nicht möglich ist. Vielfach erlebt man bei anderen informa-

tionellen Systemen, daß eine Fehlermeldung für den Benutzer nach dessen Auftragserteilung als normales Verhalten angesehen wird.

Erkundigungsdienste sollen die Möglichkeit schaffen, bereits vor Erteilung eines Auftrags zu erfahren, ob er bei tatsächlicher Erteilung erfolgreich ausgeführt werden könnte.

5.6.3 Der Storage–Bus

Im Gegensatz zum Tool–Bus, können die Telefonnummern der PEs, also deren Referenz, am Storage–Bus durch einen Auskunftsdienst ermittelt werden. Dieser Auskunftsdienst wird als PE–Ort Verwalter[1] bezeichnet und ist im ER–Diagramm dargestellt. PEs besitzen eine bisher noch nicht explizit erwähnte Eigenschaft. Sie sind mobil, d.h. sie sind in der Lage ihren Aufenthaltsort bei Bedarf in ein anderes Lager zu wechseln. Daher ist es notwendig, daß vor der Nutzung eines PE–Dienstes erst die aktuelle Telefonnummer des PEs ermittelt wird. Ist der Auftraggeber bereits in Besitz einer PE–Telefonnummer, so kann es passieren, daß diese Nummer nicht mehr gültig ist. In diesem Fall muß ebenfalls der PE–Ort–Verwalter nach der neuen Nummer befragt werden.

Die sonstige Kommunikation verläuft analog der Semantik des Tool–Bus. Man erkennt im Aufbaubild, daß es sich sowohl beim Storage–Bus als auch beim Tool– und Datentransfer–Bus um den gleichen Vermittlungsdienst handelt. Daher werden auch überall gleiche Mechanismen benutzt.

Am Storage–Bus ist keine Klassifizierung von Diensten notwendig, da PEs keine anwendungssemantischen Prüfungen machen. Die Kommunikationspartner am Storage–Bus, also Logischer Akteur und Persistenz–Akteur gelten als zuverlässig und sicher. Daher sind alle verfügbaren Dienste des Storage–Bus potentiell von allen logischen Komponenten verfügbar.

Der Storage–Bus besitzt eine weitere Fähigkeit. Er stellt eine Vermittlungsinstanz zur Verfügung, die die mit Blitzen dargestellten Broadcast/Listen–Kanäle[2] realisiert. Diese Vermittlungsinstanz ist essentieller Bestandteil des Vermittlungssystems und wird daher im Aufbaubild nicht dargestellt. Diese Vermittlungsinstanz ermöglicht es, daß sich LEs für Nachrichten bestimmter PEs anmelden können. Sendet ein PE eine Nachricht, so wird diese an alle LEs, die sich dafür angemeldet haben, verteilt[3]. So ist es möglich, daß die PEs weiter keine Kenntnis der LEs besitzen müssen.

Dieser Mechanismus kann natürlich auch von anderen Diensten benutzt werden. So kann man sich vorstellen, daß ein Dienst zum Protokollieren am Storage–Bus angeschlossen wird und alle Änderungen an PEs mitprotokolliert. Dadurch ist es möglich, das System mit beliebigen Diensten problemlos nachträglich zu erweitern. Selbst Dienste, deren Notwendigkeit man zu

[1] Der PE–Ort Verwalter wurde im Rahmen des normalen Betriebs nicht behandelt. Daher ist dieser im Aufbaubild in Abbildung 5–2 auch noch nicht vorhanden.
[2] Dies ist ein Teilnehmersystem mit Mithören ohne explizite Adressierung, d.h. einer sendet und jeder, der sich dafür interessiert, hört zu. Dies wird auch im Anhang C kurz erläutert.
[3] Das Publish/Subscribe–Verfahren wurde bereits bei den Erläuterungen zur Persistenz– und Datenverwaltungsschicht angesprochen.

Beginn des Systementwurfs noch nicht erkannt hat, können hinzugefügt werden, ohne bestehende Komponenten ändern zu müssen.

5.6.4 Der Datentransfer–Bus

Der Datentransfer–Bus unterscheidet sich von den anderen beiden Schnittstellen dadurch, daß über ihn Daten transportiert werden müssen. Sie können nicht, wie bisher, vermittelt werden. Technisch gesehen, bedeutet das, daß ein Auftrag in viele kleine aufgeteilt werden muß. Der Zustand des Auftragsfortschritts muß zusätzlich verwaltet werden. Dieser Transportdienst läßt sich nicht grundsätzlich in tiefere Protokollschichten abwälzen, da sehr oft plattformabhängige Aspekte betrachtet werden müssen. Dies kommt daher, daß es sich um riesige Datenmengen handeln kann, die nicht in einem unmittelbar verfügbaren Speicher abgelegt werden können. Dadurch müssen die beiden Transportakteure sowohl der Quelle als auch der Senke eine Möglichkeit besitzen, auf Massenspeicher in ihrem plattformspezifischen Format zugreifen zu können. Dies wird also letztlich Aufgabe der DTEs bzw. ITEs sein.

Dieser Sachverhalt läßt sich mit einem Beispiel aus dem Versandhaus vergleichen. Ein Benutzer bestellt 2000 Bücher. Der verfügbare Kurierdienst kann jedoch pro Fahrt nur 500 Bücher transportieren. Somit werden vier Fuhren notwendig sein. Diese Mengen bekommt er jedoch nicht in den Briefkasten des Empfängers. Daher ist er gezwungen, bei jeder Fuhre nach seiner Ankunft den Kunden zu benachrichtigen. Diesen muß er auffordern, ihm Zugang zu einem Ort zu verschaffen, wo er die Bücher abladen kann. Dies könnte eine Garage sein. Falls der Kunde nicht wünscht, daß der Bote Zutritt zur Garage erhält, so muß der Kunde selbst die Bücher in die Garage bringen.

Analog trifft dieser Vergleich bei heterogenen Rechnersystemen zu. Normalerweise wird man es nicht zulassen, daß ein fremder Transportakteur auf lokale Ressourcen zugreift. Dies birgt nicht nur ein Sicherheitsrisiko. Wenn der Zugriff auf Ressourcen nicht korrekt verläuft, dann könnte dies zu unerwünschten Systemfehlern führen. Im obigen Beispiel wäre das vergleichbar mit dem Einparken des Kurierwagens in der Garage. Falls der Kurierfahrer dies nicht beherrscht, kann er einen Unfall verursachen, der sowohl seinen Wagen als auch die Garage des Kunden beschädigt. Im Extremfall geht danach nichts mehr und der Vorgang muß abgebrochen werden.

5.7 Die Schaffungskomponenten

Bisher wurde fast ausschließlich über die Aufgaben der unterschiedlichen am Anwendungsbetrieb beteiligten Komponenten gesprochen. Dabei wurde mehrfach auf die Schaffungskomponenten hingewiesen, die nun vorgestellt werden. Einige der bereits bakannten Komponenten, insbesondere die zentralen Verwalter der drei Schichten, gehören einem Typ von Komponente an, welche sowohl anwendungssemantische, als auch strukturvarianzverwaltende Aufgaben übernehmen. Dabei ist es wichtig, daß die beiden „Concerns“ Betrieb und Strukturvarianzverwaltung innerhalb einer Komponente getrennt werden. Man wird bei der Realisierung eines Systems nach diesem Modell Komponenten, die mehrere „Concerns“ erfüllen

müssen, in kleinere Teilkomponenten verfeinern. Dadurch wird die Austauschbarkeit und Wiederverwendbarkeit begünstigt.

Auch in diesem Abschnitt wird ein Aufbaubild zusammen mit einem ER–Diagramm präsentiert. Dadurch ist es möglich, sowohl den Aufbau, als auch die Beziehungen, die zur Schaffung benötigt werden, besser zu verstehen. Das Aufbaubild der Schaffungskomponenten wird in Abbildung 5–4 dargestellt. Die Beziehungen zwischen den Schaffungskomponenten sind in Abbildung 5–5 abgebildet. Beim ER–Diagramm ist ein Rückblick zum Unternehmensmodell interessant, bei dem man ähnliche Strukturen finden kann. Durch die feinere Darstellung wirkt das ER–Diagramm dieses Abschnitts zwar komplexer, enthält aber prinzipiell die gleiche Struktur.

Jedes System benötigt eine Aktivierungsphase, in der alle Komponenten geschaffen werden, die zum Betrieb benötigt werden. Die Komponenten, die in dieser Phase aktiviert werden, lassen sich in zwei Kategorien unterteilen.

Die erste Kategorie besteht aus Komponenten, die durch menschliches Zutun aktiviert werden müssen. Ähnlich dem Einschalten eines Lichts per Lichtschalter, welcher durch einen Menschen betätigt werden muß, so ist es auch notwendig, daß bestimmte Systemkomponenten per Hand aktiviert werden. Der Prozeß der manuellen Aktivierung wird meistens als Bootvorgang bzw. als „Hochfahren“ bezeichnet. Dabei werden solche Komponenten, die zur Aktivierung anderer Komponenten benutzt werden, bzw. systemweit verfügbare Dienste zur Verfügung stellen, als „Well–Known“ Server bezeichnet.

Die Komponenten der zweiten Kategorie werden nicht durch einen manuellen Bootvorgang, sondern durch bereits vorhandene aktive Komponenten geschaffen, die sowohl der ersten, als auch der zweiten Kategorie angehören können. Dies kann vor, während und nach dem eigentlichen Anwendungsbetrieb stattfinden, im Gegensatz zu Komponenten der ersten Kategorie, deren Aktivierung vor Betriebsbeginn stattfinden muß.

Diese Kategorisierung kann man nur machen, wenn man sich auf die Betrachtungsebene einigt. Somit soll hier der Einschaltvorgang eines Rechners, oder das Aktivieren des Betriebssystems nicht zum Schaffungsprozeß gehören. Um eine möglichst praxisorientierte Kategorisierung zu ermöglichen, sollen hier Komponenten der ersten Kategorie als solche charakterisiert werden, die das Starten eines Betriebssystemtasks[1] zur Aktivierung voraussetzen. Komponenten der zweiten Kategorie werden innerhalb einer bereits aktivierten Komponente geschaffen und teilen sich den gemeinsamen Task. Somit hängt die Existenz von Komponenten der zweiter Kategorie direkt mit der Existenz ihrer Schaffungskomponente, deren Task sie sich teilen, zusammen.

[1] An dieser Stelle sollen zur Vereinfachung Aspekte, wie Multitasking/–threading oder Abwicklermultiplex usw. vernachlässigt werden, da sie am Prinzip nichts ändern.

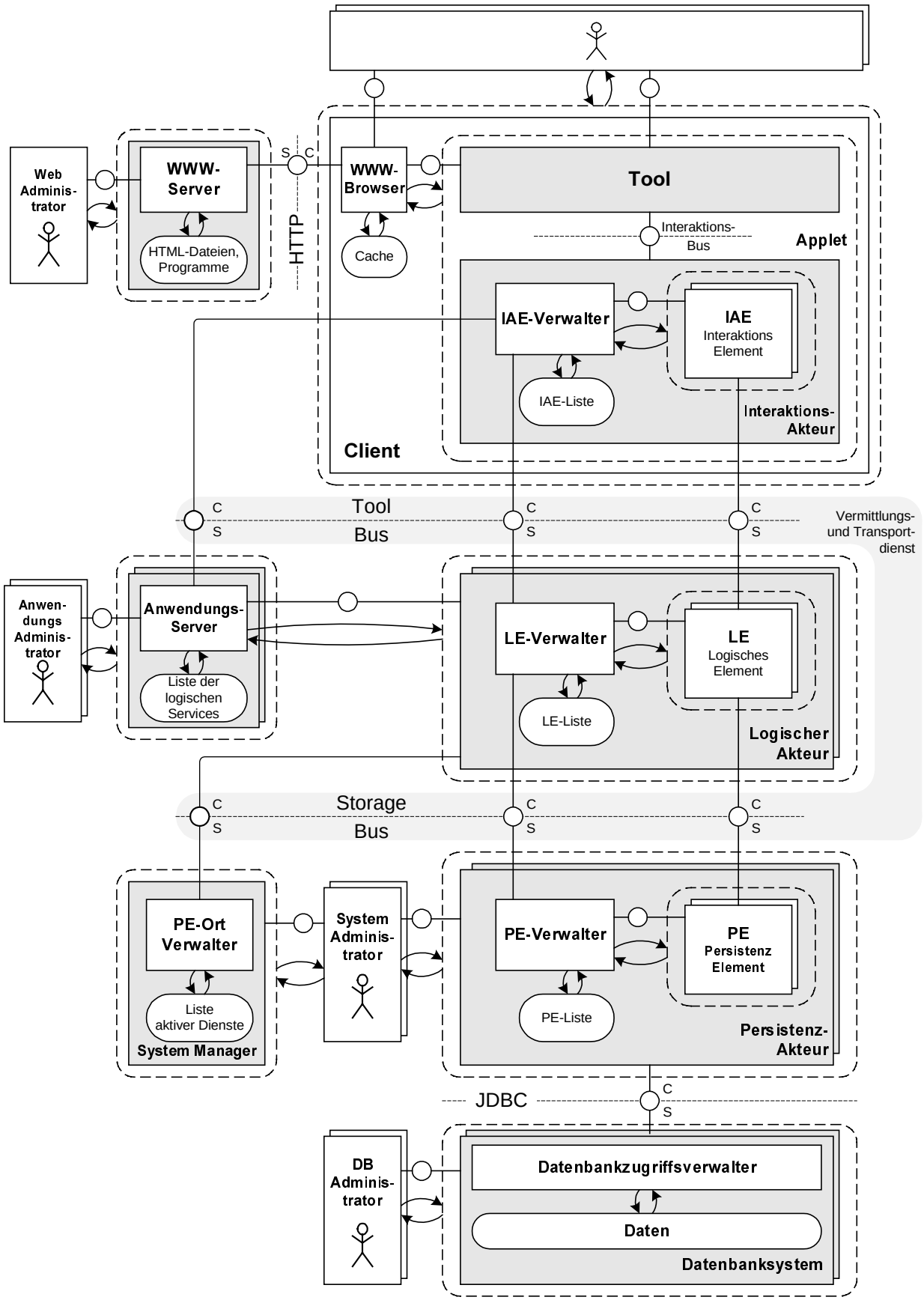

Abbildung 5–4: Aufbau der Schaffungskomponenten

5.7.1 Die „Well–Known“ Server

In Abbildung 5–4 sind mehrere Komponenten zu erkennen, die durch Menschen erzeugt werden. Davon gehören einige zur Gruppe der „Well–Known“ Server, die dadurch zu erkennen sind, daß Sie von Administratoren erzeugt bzw. aktiviert werden. Nachfolgend werden diese „Well–Known“ Server charakterisiert, damit man besser versteht, warum sie zu dieser Komponentenkategorie gehören.

- Die Datenbanksysteme

  Es wurde bereits erwähnt, daß ein System mehrere verteilte Datenlager besitzen kann. Diese werden zur Persistierung, also dauerhaften Speicherung, von Daten benutzt. Diese Datenlager werden in den meisten Fällen durch Datenbanksysteme realisiert. Diese haben bestimmte Eigenschaften, die es technisch erforderlich machen, daß der eingesetzte Abwickler nicht mit anderen Akteuren mittels Abwicklermultiplex geteilt wird. Stabilität, Zuverlässigkeit und Ausfallsicherheit sind hier einige Schlagworte. Daher besteht ein Datenbanksystem in der Regel aus einem (oder mehreren) designierten Rechnern (Abwickler) mit einem speziellen Betriebssystem. Dies macht ein Booten des Systems durch den Datenbankadministrator zu Beginn notwendig.

  Abhängig von der Größe des Gesamtsystems, können mehrere Datenbanksysteme zum Einsatz kommen. Diese müssen allerdings für deren Nutzer (Persistenz–Akteure) explizit bekannt sein und gehören daher zur Klasse der „Well–Known“–Server.

- Der System–Manager

  Diese bisher noch nicht vorgekommene Komponente dient der zentralen Informationsbeschaffung. Ihre Hauptaufgabe ist die zentrale Verwaltung des aktuellen „Orts“ von PEs. Diese besteht, analog der Telefonauskunft, darin, die Referenzen von identifizierten PEs mitzuteilen. Jeder Akteur, der lediglich die Existenz eines bestimmten PEs kennt muß, bevor er mit diesem PE kommunizieren, die Referenz (Telefonnummer) ermitteln. Diese PE–Referenz kann er durch den System–Manager in Erfahrung bringen.

  In Analogie zur Telefonauskunft muß dieser System–Manager jedem Nutzer bekannt sein und immer zur Verfügung stehen – Eigenschaften eines „Well–Known“–Servers. Beim System–Manager gibt es allerdings eine weitere Analogie zur Telefonauskunft. Wenn man die Nummer der Telefonauskunft wählt, wird man nicht immer mit der gleichen Person verbunden, sondern lediglich mit einem Dienst, der in jedem Fall das gleiche Verhalten zeigt.

  Beim System–Manager ist dieser Sachverhalt ähnlich. Prinzipiell ist die Zahl der System–Manager nicht beschränkt, eine explizite Adressierung eines bestimmten System–Managers ist jedoch nicht vorgesehen. Daher stellt der in Abbildung 5–4 dargestellte System–Manager eine logische Komponente dar, die bei der Realisierung aus vielen Akteuren mit gleichem Verhalten bestehen kann. Ein einziger Mitarbeiter in der Telefonauskunft wäre mit Tausenden von Anrufern ähnlich überfordert, wie

ein einziger System–Manager bei sehr vielen Anfragen durch viele Logische Akteure.

Bei einer Anfrage zur Ermittlung der Referenz eines PEs kann (und wird) es vorkommen, daß das gewünschte PE noch nicht „aktiv“ ist. Es wurde bereits erwähnt, daß nur eine bestimmte Anzahl Lagerarbeiter (PEs) zu einem Zeitpunkt zur Verfügung stehen kann. Damit sich die Logische Schicht mit diesem Sachverhalt nicht auseinandersetzen muß, sorgt der System–Manager dafür, daß ein gewünschtes PE aktiviert wird. Das bedeutet, daß es dem zuständigen PE–Verwalter mitteilt, daß eine bestimmte Entität im Lager benötigt wird und dafür ein Mitarbeiter (PE) abgestellt werden muß. Dadurch kann es die Referenz dieses neu geschaffenen PEs dem Anfragenden zurückliefern.

Somit ist der eigentliche Auslöser zur Aktivierung von PEs der System–Manager. Explizit werden sie jedoch vom PE–Verwalter erzeugt. Der System–Manager kann allerdings entscheiden von welchem PE–Verwalter. Dazu muß er wissen, welche Persistenz–Akteure mit welchem Datenbanksystem kommunizieren. PEs können nur von PE–Verwaltern aktiviert werden, die auf das Lager Zugriff haben, in dem die benötigten Daten des PEs abgelegt sind. Weiterhin kann der System–Manager die Wahl des PE–Verwalters von laufzeitabhängigen Faktoren machen. Dazu gehört z.B. ob der PE–Verwalter noch Kapazität für die Aktivierung zusätzlicher PEs frei hat. Ebenso kann der System–Manager die Aktivierung von PEs vom Ort des PE–Verwalters bezüglich des Anfragenden abhängig machen. Möchte beispielsweise ein LE mit seinem zugehörigen Partner–PE kommunizieren, dann ist es wünschenswert, daß dieses PE an einem Ort aktiviert wird, der für den Vermittlungsdienst am schnellsten zu erreichen ist. Somit kann der Zugriff auf PEs optimiert werden.

Der System–Manager kann allerdings noch weitere zentrale Systemdienste zur Verfügung stellen. Beispielsweise müssen vorhandene Datenbanksysteme bei ihm angemeldet werden, bevor sie den jeweiligen Persistenz–Akteuren zur Verfügung gestellt werden können. Dadurch kann die Installation von Persistenz–Akteuren erleichtert werden, da sie vom System–Manager erfahren, welches Datenlager (Datenbanksystem) ihnen zur Persistierung zur Verfügung steht.

Weiterhin ist der System–Manager fehlerredundant konzipiert um eine größtmögliche Ausfallsicherheit zu gewährleisten. Er sorgt auch dafür, daß ausgefallene Komponenten schnellstmöglich ersetzt werden, bzw. durch deren menschliche Administratoren wieder hochgefahren werden (Alarmierung bei Ausfall).

Aus den bisher beschriebenen Aufgaben des System–Managers kann auch dessen Namengebung begründet werden. Er sorgt für den optimalen Betrieb der Hauptkomponenten des Systems. Dabei kommt sehr viel Algorithmik zum Einsatz, z.B. bei einer ressourcenschonenden Aktivierung[1] von Komponenten.

[1] Dies wird meist als „Load–Balancing“ bezeichnet, da die Last möglichst gleichmäßig auf verschiedene Komponenten verteilt wird. Dies führt zu besseren Antwortzeiten und wirkt sich somit positiv auf das Laufzeitverhalten des Systems aus.

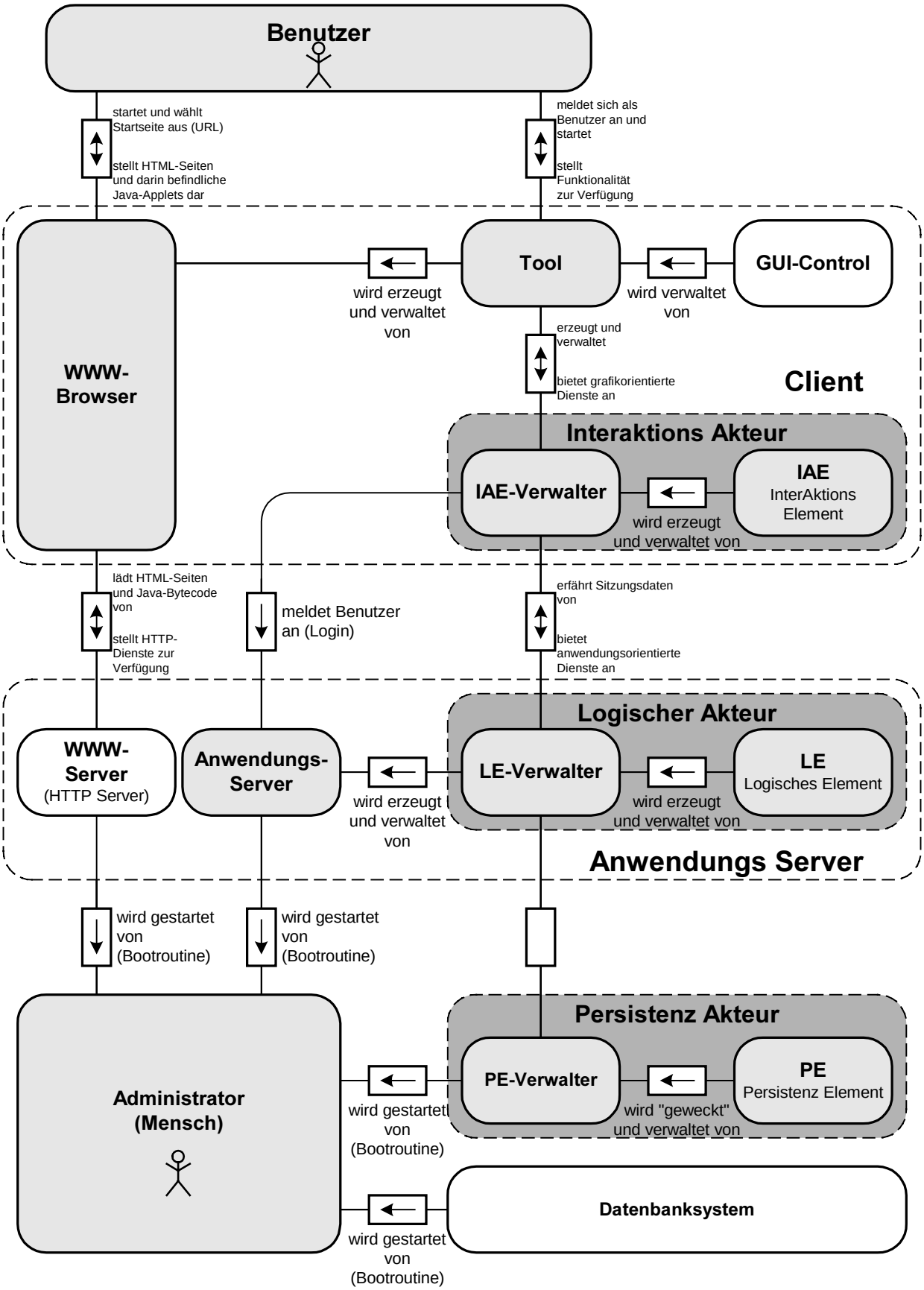

Abbildung 5–5: Beziehungen während der Schaffung von Systemkomponenten

- Die Anwendungs–Server

  Die Anwendungs–Server Komponente hat ihren Namen durch die Bezeichnung einer ähnlichen Komponente in der klassischen 3–Ebenen Architektur (3–tier architecture) erhalten. Anwendungs–Server sind die zuständigen Akteure bei dem Login und Logout–Vorgang des Benutzers. Das vom Benutzer bereits aktivierte Tool muß eine zentrale Komponente ansprechen, die einen zum Betrieb benötigten Logischen Akteur schafft.

  Hierbei gibt es zwei Realisierungsvarianten. In manchen Systemen wird die Wahl der zuständigen zentralen Komponente explizit erfolgen, d.h. daß das Tool explizit einen ganz bestimmten Anwendungs–Server auffordert, einen Logischen Akteur zu aktivieren. Die zweite Variante ermöglicht eine dynamische Auswahl des zuständigen Anwendungs–Servers im Hinblick auf Aspekte, wie Sicherheit und Lastverteilung.

  Bei der ersten Variante muß der Anwendungs–Server ein „Well–Known"–Server sein, der immer verfügbar sein muß. Das ist im Hinblick auf Ausfallsicherheit keine ideale Konstellation. Außerdem muß das Tool (möglicherweise sogar der Benutzer) den zuständigen Anwendungs Server kennen. In den meisten Systemen wird daher die zweite Variante die bessere Realisierung darstellen.

  Bei der zweiten Variante trifft eine zentrale Komponente die Auswahl des geeigneten Anwendungs–Servers. Diese Aufgabe kann im vorliegenden Entwurf z.B. vom System–Manager übernommen werden. In diesem Fall müssen sich alle Anwendungs–Server nach deren Aktivierung durch den Anwendungsadminstrator beim System–Manager anmelden. Nur durch Kenntnis der Existenz von Anwendungs–Servern, kann der System–Manager bestimmen, durch welche Anwendungs–Server die Schaffung eines Logischen Akteurs zu geschehen hat. Diese zweite Variante ist in Abbildung 5–4 nicht explizit dargestellt um eine Überladung des Bilds zu vermeiden.

  Sobald ein Interaktions–Akteur einen Anwendungs–Server kennt, wird er von diesem die Schaffung eines Logischen Akteurs in Verbindung mit einem Login des Benutzers beauftragen. Dabei muß der Anwendungs–Server Aufgaben, wie Authentifizierung des vom Interaktions–Akteurs mitgeteilten Benutzers, übernehmen. Dabei muß der Benutzer sich in irgendeiner Form ausweisen. Denn nur der, der ausgibt derjenige zu sein, der er tatsächlich ist, darf die Schaffung eines Logischen Akteurs auslösen. Dies ist ein wichtiger Aspekt, da der Logische Akteur nur durch Kenntnis seines zugehörigen Benutzers aktiviert werden kann (um z.B. juristische Berechtigungsprüfungen machen zu können).

  Der Anwendungs–Server nimmt damit die Rolle des Pförtners oder des Empfangs in einem Unternehmen ein. Nur er entscheidet wer das Unternehmen betreten darf, bzw. das System nutzen darf.

  Der Anwendungs–Server verwaltet eine Liste aller von ihm aktivierten Logischen Akteuren. Dadurch kann er Auskunft über die aktiven Benutzersitzungen geben (da ein aktivierter Logischer Akteur die Benutzersitzung auf der Logischen Schicht verwaltet). Diese Information kann z.B. für einen Systemadministrator wichtig sein.

Diese Liste ist jedoch auch wichtig zur Strukturvarianzverwaltung, denn ein Anwendungs–Server muß aktivierte Logische Akteure auch wieder deaktivieren können. Ebenso muß er den von ihm verwalteten Logischen Akteure Nachrichten senden können, um dem Benutzer Systemmeldungen mitteilen zu können (z.B. daß Benutzer wegen Wartungsarbeiten langsamere Antwortzeiten in Kauf nehmen müssen).

- Der WWW Server

  Die WWW Serverkomponente könnte auch durch andere Technologien realisiert werden als solche die direkt mit dem WWW/Internet zusammenhängen. Da allerdings abzusehen ist, daß sowohl firmeninterne als auch firmenexterne Netzwerke immer mehr durch WWW–Technologien[1] realisiert sind, wird allgemein davon ausgegangen, daß der Zugang zum System für den Benutzer nur mittels eines WWW– bzw. Webservers in Frage kommt.

  Im Unternehmensmodell muß sich der Kunde bei einer zentralen Stelle anmelden, um eine Kundenbetreuungsfiliale eingerichtet zu bekommen. Dazu muß er im Besitz der Telefonnummer dieser zentralen Stelle sein. Diese Nummer kann er von diversen Quellen erfahren haben (Werbung, Hörensagen usw.).

  Beim WWW–Server ist es ähnlich. Die Telefonnummern des Internet werden URLs genannt. Um den Zugang zum System zu erlangen, muß er die URL des WWW–Servers kennen. Dadurch kann er mittels seines WWW–Browsers ein Benutzersitzung beginnen. Dies setzt allerdings das Vorhandensein des entsprechenden WWW–Servers voraus, weshalb dieser zur Klasse der „Well–Known“–Server gehört.
  Der WWW–Server liefert sowohl die HTML–Seite, die als Einstieg zum System dient, als auch die Verhaltensbeschreibung (Programmcode) des Applets mittels des Standardprotokolls HTTP. Die Terminologie ist an dieser Stelle sehr Technologie–lastig. Prinzipiell sind diese Komponente auch in ähnlicher Form durch andere Technologien realisierbar. Da an dieser Stelle keine technologischen Details angesprochen werden sollen, befindet sich über das Zusammenspiel der WWW–Komponenten in Anhang B weiterführende Information.

5.7.2 Der Persistenz–Akteur

Ähnlich den Datenbanksystemen, kann es auch viele Persistenz–Akteure geben. In der Regel wird es mehr Persistenz–Akteure geben als Datenbanksysteme. Dies liegt zum einen darin begründet, daß ein Persistenz–Akteur technisch nicht die Komplexität besitzen wird, wie ein Datenbanksystem. Zum anderen wird die Skalierbarkeit des Systems erhöht, wenn je nach Bedarf beliebig viele Persistenz–Akteure aktiviert werden können.

Allerdings gehört der Persistenz–Akteur nicht zur Klasse der „Well–Known“–Servern. Vielmehr muß sich ein vom Systemadministrator aktivierter Persistenz–Akteur beim System–Manager anmelden, bevor er zum Betrieb zur Verfügung steht. Dies ist vergleichbar mit der

[1] Firmeninterne TCP/IP–Netzwerke werden auch als Intranet, für bestimmte firmenexterne Benutzer werden sie auch als Extranet bezeichnet. Diese Begriffe leiten sich aus dem Oberbegriff Internet ab.

zweiten Variante der Anwendungs–Server, die sich ebenfalls nach der Aktivierung beim System–Manager angemeldet haben. Persistenz–Akteure werden vom Systemadministrator sowohl aktiviert, als auch deaktiviert. Vor der Deaktivierung sollte sich ein Persistenz–Akteur beim System–Manager ordnungsgemäß abmelden. Falls dies nicht mehr kontrolliert möglich ist, müssen entsprechende Maßnahmen vom System–Manager und/oder vom Systemadministrator getroffen werden.

Der Persistenz–Akteur besitzt als zentralen Verwalter einen PE–Verwalter. Dieser übernimmt zentrale Aufgaben, die getrennt behandelt werden.

5.7.3 Der WWW–Browser

Auch hier gilt das bereits beim WWW–Server genannte bezüglich der technologischen Namengebung. Der WWW–Browser wird vom Benutzer aktiviert. WWW–Browser sind so allgemein verbreitet, daß sie für diese Systemklasse eine ideale technische Plattform für die Client Schicht darstellen. Der WWW–Browser muß lediglich zwei Aufgaben übernehmen. Eine davon ist es den Zugang zur HTML–Einstiegsseite des Systems zu ermöglichen. Die andere Aufgabe ist es, einen geeigneten Abwickler (Interpreter) zur Verfügung zu stellen, der die Verhaltensbeschreibung von Tool und Interaktions–Akteur interpretieren kann. Die Abwicklung dieser beiden Akteure erfolgt innerhalb des in Abbildung 5–4 zu erkennenden Applets. Auch hier sei auf Anhang B verwiesen, der die konkreten Aspekte dieser Technologie kurz behandelt.

5.7.4 Die Zentralverwalter der drei Schichten

Das in Abbildung 5–4 dargestellte Aufbaubild als auch das in Abbildung 5–5 dargestellte ER–Diagramm lassen eine starke Symmetrie der drei Hauptakteure Interaktions–, Logischer und Persistenz–Akteur erkennen. Diese Symmetrie zeichnet sich durch das Vorhandensein eines Verwalters vieler Elemente aus. Diese Symmetrie kann bereits im Unternehmensmodell begründet werden. Es handelt sich dabei um drei eigenständige Akteure(Unternehmensteile), die jeweils gleichartige Elemente (Mitarbeiter) einsetzen, um die semantischen Aufgaben der jeweiligen Schicht zu erfüllen. Um eine optimale Verwaltung dieser Elemente zu ermöglichen, ist ein eigens dafür zuständiger Verwalter (Personalabteilung) vorhanden.

Da sich diese Verwalter nur im Detail unterscheiden, werden zuerst schichtübergreifend die allgemeinen Aufgaben aller Verwalter genannt. Danach erfolgt eine Behandlung der speziellen schichtabhängigen Aufgaben.

Jeder der drei Zentralverwalter besitzt als Hauptaufgabe die Strukturvarianzverwaltung der durch ihn erzeugten Elemente (IAE, LE bzw. PE). Dabei besitzt jeder der Verwalter eine Liste, in der vermerkt wird, welche Elemente bereits aktiviert wurden. Dies ist nicht nur zur Aktivierung und Deaktivierung der Elemente notwendig, sondern wird auch während des Betriebs benötigt. Es ist nämlich wünschenswert, daß die Kommunikation der Elemente untereinander nicht von deren Aktivierung abhängt. So ist es möglich, daß ein Element durch Dienste des Verwalters mit anderen identifizierbaren Elementen reden kann, ohne zu wissen, ob sie bereits durch den Verwalter aktiviert wurden. Ein Element identifiziert ein anderes

Element gegenüber seinem Verwalter. Dieser prüft, ob das identifizierte Element bereits durch ihn aktiviert wurde. Falls es noch nicht aktiviert wurde, wird dies zu diesem Zeitpunkt durch den Verwalter erledigt. Danach wird die Referenz (Zeiger) an das fragende Element zurückgeliefert. Somit stellen die Verwalter schichtinterne Auskunftsdienste zur Verfügung, die eine Auflösung von Identifikatoren in Zeiger (Referenzen) möglich macht. Dies kann eine Erzeugung von Elementen erforderlich machen.

Die jeweiligen Verwalter sind ebenso zuständig, Daten zu verwalten, die im Rahmen der Sitzung benötigt werden. Zum Beispiel ist die Anzahl aktivierbarer LEs beschränkt. Ein LE–Verwalter muß entscheiden, welche LEs weiterhin benötigt werden und welche wieder deaktiviert bzw. gelöscht werden können. Dazu muß er wissen welche Ressourcen ihm zur Verfügung stehen. Analog der Realität, kann eine Personalabteilung nur soviele Mitarbeiter einstellen, die in den vorhanden Räumlichkeiten Platz finden. Dazu muß die Personalabteilung die Räumlichkeiten kennen.

Wenn ein Element mit seinem Partnerelement der Schicht darunter kommunizieren möchte, ist es von der Vermittlungshilfe anderer angewiesen. Es wurde bereits in früheren Abschnitten erwähnt, daß der Vermittlungsdienst für Elemente transparent bleiben soll. Ein Element kommuniziert mit seinem Partner grundsätzlich indirekt mittels seines Verwalters. Als Analogie zur Realität könnte man sagen, daß sich ein Mitarbeiter eines Unternehmens bevor er mit jemandem kommunizieren möchte, sich durch sein Sekretariat mit seinem Gesprächspartner verbinden läßt. Dazu muß er der Sekretärin den Namen des Gesprächspartners nennen. Diese ermittelt die Telefonnummer und den jeweiligen Fernmeldedienst und stellt die Verbindung her. Die aufgebaute Verbindung wird an den Mitarbeiter zur Nutzung weitergegeben.

Auch hier werden die jeweiligen Elemente bevor sie Aufträge an ihren Partner senden, eine Verbindung durch ihren Verwalter herstellen lassen. Der Verwalter besitzt zur Ermittlung der Verbindungsdaten einige Freiheiten. So kann er sich beispielsweise die Referenz (Telefonnummer) des gewünschten Partners merken. Falls bei späteren Versuchen keine Verbindung unter der gespeicherten Referenz mehr möglich ist, wird durch die jeweilige Auskunft eine neue besorgt. Somit sind die Verwalter auch mit der Aufgabe betraut, die zur Verfügung stehenden Vermittlungsdienste zu kennen und ihre Funktionsweise zu verstehen.

- Der IAE–Verwalter

  Die Besonderheit des IAE–Verwalters besteht in der Fähigkeit, implizit die Erzeugung des zugehörigen Logischen Akteurs zu veranlassen. Dabei muß der Benutzer, der die Aktivierung des IAE–Verwalters durch das Applet veranlaßt hat, sich ausweisen. Dadurch kann der IAE–Verwalter eine Authentifizierung durch einen Anwendungs–Server veranlassen. Dieser Vorgang ist der Login–Prozeß. Bei erfolgreichem Login, wird ein Logischer Akteur erzeugt. Die Referenz dessen LE–Verwalters wird dem IAE–Verwalter zurückgeliefert.

  Damit IAEs mit LEs reden können, ist es grundsätzlich erforderlich, daß diese von deren Existenz wissen. Zu Beginn gibt es jedoch weder IAEs noch LEs. Daher muß das sogenannte Wurzel–LE vom IAE–Verwalter in Erfahrung gebracht werden. Dies erfährt er durch einen Auskunftsdienst des neu erzeugten LE–Verwalters. Danach wird die Existenz der unterschiedlichen LEs durch bereits vorhandene LEs mitgeteilt.

Dieses Prinzip des Hörensagens wurde bereits bei der Beschreibung des Tool–Bus im Abschnitt 5.6.2 angesprochen.

Während des Betriebs kann der IAE–Verwalter jederzeit erneut das Wurzel–LE als Dienst für die von ihm verwalteten IAEs in Erfahrung bringen. Weiterhin kann durch den IAE–Verwalter der Deaktivierungsvorgang eingeleitet werden. Dazu bietet er dem Tool einen Logout–Dienst an. Dabei teilt er dem LE–Verwalter mit, daß er ihn nicht mehr benötigt. Danach werden die IAEs vom IAE–Verwalter gelöscht, der selbst wiederum nur durch den WWW–Browser (Benutzer) deaktiviert werden kann.

- Der LE–Verwalter

  Im Gegensatz zum IAE–Verwalter, besitzt der LE–Verwalter nicht nur einen Ansprechpartner auf der für ihn tieferen Schicht, der Persistenz–Schicht. Unter Umständen muß der LE–Verwalter mit sehr vielen PE–Verwaltern in Kommunikation treten. Allerdings muß er sich weder deren Referenz noch deren Identität merken. Er muß lediglich wissen, daß es Akteure der Klasse PE–Verwalter gibt, um deren Dienste in Anspruch zu nehmen.

  Der eigentliche Ansprechpartner des LE–Verwalter ist der System–Manager. Von ihm erfährt er, wo sich von ihm identifizierbare PEs befinden. Dieser Ort liefert bereits die direkte Referenz des PEs, so daß ein LE–Verwalter für diese Aufgabe den PE–Verwalter gar nicht kennen muß.

  Der LE–Verwalter bietet der Interaktions–Schicht darüber hinaus die bereits angesprochenen Dienste bezüglich der aktuellen logischen Benutzersitzung an. Diese umfassen z.B. die Auskunft über die Existenz des Wurzel–LEs. Aber auch sehr allgemeine benutzerunabhängige Dienste, wie z.B. die Systemzeit können vom LE–Verwalter ermittelt werden. Diese Dienste sind insbesondere innerhalb des Logischen Akteurs von Bedeutung, denn so ist es möglich zentrale Dienste allen LEs anzubieten. Die Anbindung eines Berechtigungswesens kann so z.B. zentral erfolgen. Wie ein LE–Verwalter die Information ermittelt, die ein LE benötigt um eine juristische Prüfung durchzuführen, ist für das LE nicht wichtig. Wichtig ist jedoch, daß der LE–Verwalter diesen Dienst für das LE anbietet. So ist jederzeit ein Wechsel des Berechtigungswesens möglich, ohne Einfluß auf das Verhalten der LEs zu haben.

  Der LE–Verwalter stellt auch einige schichtinterne Dienste den von ihm verwalteten LEs zur Verfügung. So kann er beispielsweise die Verbindung zu PEs aufbauen, bzw. vermitteln. LEs können ihrem LE–Verwalter aber auch mitteilen, daß sie nicht länger benötigt werden. Daraufhin werden sie vom LE–Verwalter mitsamt ihrem Eintrag in der LE–Liste gelöscht. In der anderen Richtung können sie benötigte LEs erzeugen. Diese, bereits in der Einleitung dieses Abschnitts angesprochene, Aufgabe ist sowohl schichtintern durch die LEs als auch seitens des IAE–Verwalters möglich.

  Die letzte Gruppe von Diensten, sind die, die für den Anwendungs–Server zur Verfügung gestellt werden müssen. Zum Beispiel muß der Anwendungs–Server erfahren können, wie viele Ressourcen der Logische Akteur derzeit benötigt. Nur so kann er entscheiden, wie viele Ressourcen übrig bleiben, um die Erzeugung weiterer Logi-

scher Akteure zuzulassen. Ebenso muß der LE–Verwalter Mitteilungen vom Anwendungs–Server behandeln können. Dazu gehört z.B. die Nachricht, daß die Ressourcen knapp werden und der LE–Verwalter nicht benötigte LEs möglichst deaktivieren sollte.

Die Menge dieser vielfältigen Dienste, die verschiedene „Concerns“ betreffen, ist relativ groß. Daher ist eine verfeinerte Struktur innerhalb des LE–Verwalters notwendig. Diese ermöglicht eine Trennung der unterschiedlichen „Concerns“. Diese verfeinerte Struktur geht allerdings über den Inhalt dieser Diplomarbeit hinaus und hängt vom letztlich zu realisierenden System ab.

- Der PE–Verwalter

  Der PE–Verwalter unterscheidet sich im wesentlichen in zwei Punkten von den anderen beiden Verwaltern. Zum einen werden PEs nicht erzeugt und gelöscht, sondern bereits vorhandene werden „geweckt“ und „schlafen gelegt“ und noch nicht vorhandene werden erzeugt (und endgültig gelöscht). Zum anderen müssen erzeugte sowie geweckte PEs beim System–Manager angemeldet werden, damit dieser ihren Status kennt.

  PEs stellen Lagerarbeiter dar, die immer für einen bestimmten granularen Datenbereich des Lagers zuständig sind. Die Existenz eines solchen Datenbereichs bedeutet, daß es auch einen Lagerarbeiter geben kann, der für diesen Bereich zuständig ist. Dabei können allerdings nicht alle PEs zum gleichen Zeitpunkt existieren. Dies macht auch keinen Sinn, da zu einem Zeitpunkt auch nur ein kleiner Bruchteil des Lagerbestands benötigt wird. Daher soll hier die Vorstellung gelten, daß jedem gelagerten Granulat ein Lagerarbeiter (PE) zugeordnet ist. PEs werden aber nur dann aktiviert, wenn sie benötigt werden. Ein solcher Vorgang wird mit dem „Wecken“ von PEs bezeichnet. Analog dazu werden sie „schlafen gelegt“, wenn sie nicht mehr benötigt werden.

  Ein solches PE kann von jedem PE–Verwalter geweckt werden, der auch den Lagerbereich kennt, für den das PE zuständig ist. Es wurde bereits erwähnt, daß der System–Manager entscheidet, welcher PE–Verwalter ein PE wecken soll. Daher muß der System–Manager wissen, welche PE–Verwalter auf welche Datenlager (Datenbanksysteme) zugreifen können. Von diesen sucht er sich den optimalen aus.

  Der ausgesuchte PE–Verwalter weckt das PE. Dies geschieht dadurch, daß es sich ein PE der entsprechenden Klasse erzeugt, und diesem mitteilt, für welchen Lagerbereich es zuständig ist. Dazu muß der PE–Verwalter das zu benutzende Datenbanksystem kennen. Diese Mitteilung des zugeordneten Datenbereichs erfolgt implizit mittels Umschreibung bzw. Identifikation. Dadurch muß daß PE nicht wissen wie seine Daten in der Datenbank technisch bedingt abgelegt werden.

  Hier muß jedoch gesagt werden, daß dies ein stark technologieabhängiger Aspekt darstellt. Die eingesetzte Datenbanktechnologie innerhalb eines Persistenz–Akteurs sollte im Vorfeld festgelegt sein. Zur Unterstützung mehrerer unterschiedlicher Datenbanktechnologien, wird man verschieden Typen von Persistenz–Akteuren realisie-

ren, die jeweils optimal auf die für sie zutreffende DB–Technologie zugeschnitten sind. Bei einer relationalen Datenbank mit einer genormten deskriptiven Zugriffssprache, wie SQL beispielsweise, muß ein PE formulieren, welche Daten benötigt werden. Der PE–Verwalter kann hier allenfalls die Verbindung zur Datenbank selbst kapseln, nicht jedoch den Zugriff. Dies hat als Konsequenz, daß das PE aus zwei Teilen bestehen muß: einem datenbankabhängigen und einem datenbankunabhängigen Teil. Nur so ist der Austausch von Datenbanktechnologie zu erreichen.

Beim Wecken bzw. Erzeugen von PEs wird deren Existenz und Status dem System–Manager mitgeteilt. Dadurch kann er die Referenz „geweckter“ PEs mitteilen.

Eine weitere Aufgabe liegt im Bereich des PE–Verwalters. Dieses Konzept ist allerdings noch im Vorentwicklungsstadion und soll daher nur kurz angedeutet werden. Es sieht den Transport von PEs innerhalb der Persistenz Schicht vor. Das bedeutet, daß ein PE mit allen seinen Daten von einem Persistenz–Akteur in den Bereich eines anderen Persistenz–Akteurs „wandern“ kann. Dies wird sinnvoll sein, um eine räumliche Nähe der Daten zur Anwendung, welche die entsprechenden PEs nutzt, schaffen zu können.

Als Beispiel seien alle PEs der Dokumente in einem Büro eines Benutzers. Die Daten dieser PEs liegen in einer Datenbank, die sich räumlich in der Nähe des Benutzers befindet. Dabei sind optimale Antwortzeiten zu erwarten. Sobald dieser Benutzer allerdings zu einem Unternehmensstützpunkt in einem anderen Land versetzt wird, müssen die Daten bei jeder Nutzung vom alten Ort zum Neuen transportiert werden. Steht an diesem neuen Stützpunkt ebenfalls die Infrastruktur (Datenbanksystem, Persistenz–Akteur, Anwendungs–Server), dann macht es Sinn, die ausschließlich von ihm benutzten Daten an diesen Ort zu transportieren. Dies kann sowohl durch einen impliziten Anstoß des Benutzers geschehen, der durch entsprechende Dienste den gewünschten Ort bestimmter Daten selbst festlegt. Dies kann allerdings auch durch eine Algorithmik geschehen, die feststellt, welche Daten wo am häufigsten benötigt werden.

In beiden Fällen wird der Anstoß eines solchen Transports vom System–Manager kommen, der einen PE–Verwalter im Bereich des alten Lagerorts der Daten beauftragt, diese zu einem anderen PE–Verwalter zu transportieren. Dabei wird sowohl der Zustand (Wertbelegung) als auch der Identifikator an den neuen Ort transportiert, wo ein „leeres“, bzw. zustandsloses PEs mit diesen Daten initialisiert wird. Nach Abschluß dieses Vorgangs, wird das alte PE endgültig gelöscht. Das neue kann schlafen gelegt werden.

Im Bereich der Persistenz Schicht gilt es noch einige Konzepte auszuarbeiten, die zu diesem Zeitpunkt noch nicht den Fortschritt erreicht haben, als daß sie im Rahmen dieser Diplomarbeit erwähnt werden könnten. Dennoch bietet dieses Architekturkonzept einige Möglichkeiten, das schwierige Problem der Verteilung von Daten anzugehen.

5.8 Zustandsveränderungen im System

Bis zu diesem Zeitpunkt war immer nur von einem System die Rede, welches ausschließlich einen lesenden Zugriff auf Daten ermöglicht hat. An manchen Stellen wurde dennoch angedeutet, daß Änderungen des Systemzustands möglich sind. Aus Gründen der besseren Verständlichkeit, mußte auf die Erläuterungen der Semantik der Zustandsveränderung bisher verzichtet werden.

Änderungen von Daten in einem Mehr–Benutzer–Szenario führt grundsätzlich zur Problematik der konkurrierenden Zugriffe. Diese machen es erforderlich, daß Datenbereiche zur exklusiven Nutzung gesperrt und erst nach der Zustandsänderung wieder freigegeben werden können. Bei der Diskussion der Granularität von PEs (Abschnitt 5.5.3) wurde bereits angesprochen, daß oft nicht die Datenbereiche gesperrt werden, die von einer Änderung betroffen sind. Häufig werden aus Performance–Gründen andere Granulate gesperrt (z.B. ganze Datenbank–Bereiche oder Seiten).

Bei dem bisher vorgestellten Architekturkonzept, können konkurrierende Zugriffe ausschließlich bei den PEs zu Problemen führen. Diese stellen jedoch die bereits besprochenen elementaren Granulate des Systems dar. Das bedeutet, daß eine Änderung eines PEs dazu führt, daß dieses gesperrt werden muß. Nur so kann man verhindern, daß der gleichzeitig stattfindende Zugriffsversuch eines anderen (LEs) das PE in einen anwendungssemantisch inkonsistenten Zustand überführt. Dies liegt darin begründet, daß nicht jede Änderung eines PEs aus einer elementaren Operation besteht, sondern häufig aus mehreren Änderungsoperationen. Der anfangs konsistente Zustand, wird nur dann in einen konsistenten überführt, wenn alle Änderungsoperationen durchgeführt werden.

Diese Vorstellung einer elementaren Operation, die aus mehreren Teiloperationen besteht, impliziert eine Transaktion, deren Semantik genau dieser Atomaritätsforderung entspricht. Die Frage ist nur, ob die Transaktion in ihrer üblichen Form an dieser Stelle das geeignete Mittel ist. Bei der Zustandsänderung nur eines einzigen PEs scheint dies kein Problem darzustellen. Sobald aber mehrere PEs an einer zustandsverändernden Operation beteiligt sind, dürfte das Problem klar werden.

Das Problem an diesem Ansatz wird durch die Verteilbarkeit der verschiedenen Komponenten der Persistenz Schicht ausgelöst. Man stelle sich vor, an einer Änderungsoperation seien zwei PEs beteiligt, die unterschiedlichen Persistenz–Akteuren angehören. Nur die Änderung beider PEs gilt als korrekte Überführung eines konsistenten Zustands in einen neuen konsistenten Zustand. Sollten die Änderungen eines PEs erfolgreich sein, die des anderen jedoch nicht (z.B. wegen Netzwerkfehler), dann wird man große Schwierigkeiten haben, ein Transaktionskonzept zu finden, welches den alten konsistenten Zustand wieder herstellen kann. Auf Transaktionskonzepte soll hier jedoch nicht näher eingegangen werden.

Das hier vorgestellte Architekturkonzept versucht einen anderen Weg zu gehen. Es handelt sich dabei um ein transaktionsähnliches Konzept, unterscheidet sich jedoch zum einen durch die Namengebung, als auch durch den Zwang, Performance in den Vordergrund zu stellen.

5.8.1 Das Bauzaun Modell

Es wurde bereits angesprochen, daß ein PE von mehreren LEs genutzt werden kann. Dies setzt jedoch voraus, daß keines der LEs zustandsverändernde Dienste des PEs in Anspruch nimmt. An dieser Stelle ist eine Kategorisierung der Dienste eines PEs notwendig, damit man die zustandsverändernden Dienste von den allgemeinen Diensten unterscheiden kann.

- Allgemeine Dienste

  Unter allgemeinen Diensten werden zwei verschiedene Dienste verstanden. Zum einen sind dies die reinen Auskunftsdienste, wie z.B. die Ermittlung der Farbe eines Regals. Zum anderen bestehen allgemeine Dienste aus jenen Diensten, die zwar den Zustand eines PEs verändern, die jedoch keine Sperrung des PEs benötigen und somit nicht zu Konflikten oder Problemen führen. Ein Beispiel eines solchen Dienstes ist das Anbringen eines Kommentars am Regal. Potentiell können unendlich viele Kommentare zum gleichen Zeitpunkt an das Regal gehängt werden, ohne dafür exklusiv das Regal–PE nutzen können zu müssen. Solche Aufträge werden durch den Vermittlungsdienst sowie durch das Trägersystem des PEs automatisch sequentialisiert, bzw. hintereinander angeliefert. Sperren zur Sequentialisierung von normalen Diensten werden nicht benötigt. Bei herkömmlichen Transaktionsverwaltungssystemen ist dies jedoch ein wichtiger Faktor.

- Zustandsverändernde Dienste

  Dienste dieser Kategorie setzen den exklusiven Zugriff auf ein PE durch einen einzigen Auftraggeber voraus. Die Änderung der Reihenfolge der Bretter des im obigen Beispiel genannten Regals erfordert die exklusive Nutzung des Regals. Kein anderer als der Auftraggeber darf bis zum Ende aller Änderungen zustandsverändernde Dienste in Anspruch nehmen. Unter bestimmten Voraussetzungen wird die Nutzung eines Teils der allgemeinen Dienste möglich bleiben. Bei Änderung der Brettreihenfolge sollte ein Erfragen der Regalfarbe weiterhin möglich bleiben. Dies hängt jedoch von der Komplexität des PEs ab und kann im Einzelfall den Erfordernissen angepaßt werden.

Das hier vorgestellte Bauzaun Modell erhielt seinen Namen durch Überlegungen, die eng an die Semantik von TimeLess angeknüpft sind. Der Begriff „Bauzaun“ bezeichnet eine Absperrung um Gebäude, die sich im Bau befinden. Dies kommt daher, daß auch in TimeLess die Entität „Gebäude“ vorkommt. Am Beispiel eines Gebäudes soll nun das Sperrkonzept dieses Entwurfs dargestellt werden.

Ein Bauunternehmen möchte gewisse Änderungen (Renovierung) an einem Gebäude durchführen. In Analogie zur Realität, kann er bestimmte Änderungen an einem Gebäude nur vornehmen, wenn alle Menschen das Gebäude verlassen haben. Daher stellt er einen Bauzaun um das Gebäude, der es erlaubt das Gebäude zu verlassen aber nicht mehr es zu betreten. Nach einer Sperrzeit, bzw. nachdem alle das Gebäude verlassen haben, wird das Gebäude für Änderungen freigegeben. Eine andere Möglichkeit besteht darin, das Gebäude nicht komplett zu sperren, sondern lediglich alle Menschen, die das Gebäude betreten, auf eventuell stattfinden-

de Veränderungen hinzuweisen. Dadurch kann es vorkommen, daß das Betreten des Gebäudes zwar möglich ist, aber nur mit Vorsicht zu genießen ist.

Dieses Konzept wird nun auf ein Gebäude–PE übertragen. Ein Bibliothekar möchte Änderungen an einem in TimeLess befindlichen Gebäude machen. Dabei möchte er dem Gebäude eine neue Etage hinzufügen. Er (der Bibliothekar!) stellt dazu einen Bauzaun um das Gebäude, welches den Eintritt in das Gebäude zwar ermöglicht, aber mit dem Hinweis an andere versehen, daß sie eventuell mit Änderungen rechnen müssen. Die neue Etage wird nach deren Erzeugung mit einem Bauzaun versehen, der keinem anderen Benutzer Zutritt gewährt. Somit kann der Bibliothekar in aller Ruhe seine neue Etage mit Räumen und Regalen bestücken, ohne Störungen in Kauf nehmen zu müssen. Dieser Vorgang kann sich sogar über mehrere Sitzungen hinaus hinziehen, da der Bauzaun zum persistenten Zustand von PEs gehört. Sobald er mit den Änderungen am Ende ist, entfernt der Bibliothekar den Bauzaun. Danach ist die Nutzung durch andere Bibliotheksbenutzer ohne weiteres möglich (sofern sie die Berechtigung besitzen).

Dieses Konzept ist letztendlich ein pessimistisches[1] Sperrverfahren verbunden mit langen, also sitzungsüberdauernde Transaktionen. Vielfach wird das pessimistische Sperren nicht als Alternative zum optimistischen Sperrverfahren angesehen, da es bei Systemen mit sehr großer Änderungshäufigkeit zu Performanceproblemen führen kann. Allerdings kann das optimistische Sperrverfahren bei verteilten Systemen nicht eingesetzt werden, da das bereits angesprochene Problem, bei Fehlern alle Datenänderungen „zurückzurollen“ nur durch sehr hohen Aufwand betrieben werden kann – in vielen Fällen ist dies gar nicht möglich. Selbst durch den Einsatz spezieller Protokolle[2] ist dieses Problem bei verteilten dezentralen Systemen sehr problematisch. Daraus folgt, daß das pessimistische Verfahren das einzig in Frage kommende Sperrverfahren darstellt.

Die oft vorhandenen Performanceproblematik des pessimistischen Sperrens hängt jedoch damit zusammen, daß die zu sperrenden Granulate nicht der Semantik der Anwendung angepaßt sind. Dieses Problem läßt sich allerdings durch die optimale Granularität der PEs bewältigen (siehe Abschnitt 5.5.3).

Weiterhin bietet das Bauzaun Modell den Vorteil, daß andere Benutzer zu jedem Zeitpunkt über Änderungen in Kenntnis gesetzt werden können. Somit sind sie nicht verwundert, wenn sie selbst nicht in der Lage sind Änderungen vorzunehmen. Bei einer optimalen Granularisierung der PEs wird in den meisten Fällen ein konkurrierender Zugriff die Ausnahmesituation bleiben.

---

[1] Beim pessimistischen Sperren werden Sperren vor der Datenänderung besorgt. Erst wenn alle Sperren vorhanden sind, wird die Datenänderung durchgeführt. Beim optimistischen Sperren, wird beim Versuch Daten zu ändern der betroffene Bereich erst gesperrt. Ist das Sperren zu diesem Zeitpunkt nicht möglich (falls bereits gesperrt ist), dann werden alle damit verbundenen Änderungen abgebrochen und der Ausgangszustand wird wieder hergestellt. Wegen des höheren Verwaltungsaufwands und dem damit verbundenen Performanceproblem wird normalerweise das optimistische Sperren bevorzugt eingesetzt.

[2] Stellvertretend soll hier das „Two–Phase–Commit“ genannt werden. Ähnlich einem vier–Flanken Handshake, werden dabei zuerst alle Änderungen beauftragt. Erst bei Bestätigung der erfolgreichen Durchführung aller Änderungen, wird ein final commit, bzw. eine Abschlußerlaubnis an alle geänderten Stellen gegeben. Das Problem dieses Protokolls liegt in dem Mißlingen des final commits. Wenn nur ein Teil der geänderten Akteure von dem final–commit erfährt, kann das System in einen inkonsistenten Zustand überführt werden.

5.8.2 Änderungen von Daten

Änderungen des vorhandenen überdauernden, bzw. persistenten Systemzustands werden mittels PEs vorgenommen. Persistenz Elemente haben ihren Namen nicht erhalten, weil sie per Definition persistent sind, sondern weil sie für Persistenz sorgen. Das heißt, sie garantieren, daß an Ihnen vorgenommene Zustandsänderungen dauerhaft sein werden, egal was passiert. Dies schließt technische Probleme und sonstige unvorhersehbare Störungen mit ein.

Bei erfolgreicher Änderung eines PEs, werden alle zuhörenden Abonnenten, mittels des bereits vorgestellten Nachrichtenkanals (Blitz), benachrichtigt, daß eine Änderung stattgefunden hat. Dadurch können die Abonnenten (in der Regel LEs) bestimmte Operationen durchführen, wie z.B. dem Benutzer diese Änderung mitteilen. Eine vollständige Spezifikation der Änderungssemantik soll jedoch nicht Thema dieser Arbeit sein.

Kapitel 6 Zusammenfassung/Ausblick

Basierend auf dem in Kapitel 5 vorgestellten Architekturkonzept wurde im Rahmen des TimeLess-Projekts in nunmehr vier Entwicklungsinkrementen erfolgreich ein funktionsfähiger Prototyp entwickelt. Die Entwicklung wurde anhand des in Kapitel 2 vorgestellten Prozeßmodells durchgeführt. Dabei wurde festgestellt, daß das inkrementelle Prototyping das ideale Entwicklungsmodell darstellt für das in dieser Arbeit erläuterte TimeLess–Abteilungsprodukt.

Insbesondere die architekturellen Konzepte, die in dieser Arbeit vorgestellt wurden, konnten erfolgreich umgesetzt werden. Dabei konnten alle anfallenden Probleme im Rahmen des Architekturmodells gelöst werden. Die in Kapitel 3, Abschnitt 3.3.1 eingeschränkten Anforderungen konnten fast vollständig umgesetzt werden.

Das in Kapitel 4 vorgestellte Unternehmensmodell dient nicht nur zur Beschreibung der in dieser Arbeit vorgestellten Konzepte, sondern dient im Labor als Modell zur Schaffung von Verständnis. Die Nähe des hier vorgestellten Architekturkonzepts zur Unternehmensrealität verdeutlicht, daß der in der Informatik oft gegangene Weg der datenzentrischen Modellbildung nicht immer ein geeignetes Mittel des Systementwurfs sein muß. Insbesondere in größeren Projekten, in denen eine hocheffiziente, unmißverständliche Kommunikation unter vielen Entwicklern große Bedeutung hat, kann ein realitätsnahes Modell ausschlaggebend für den Erfolg sein.

6.1 Ausblick

Bedingt durch den allgemeinen Charakter dieses Architekturkonzepts sind eine große Vielfalt unterschiedlichster Anwendungen realisierbar. Insbesondere die Verteilung von Daten und anwendungssemantischen Granularisierung sind Konzepte, die insbesondere die EDV-Welt großer Unternehmen positiv verändern könnten. Die Schaffung einer unternehmensweiten dezentralen Datenhaltungsinfrastruktur mit allen benötigten Eigenschaften, wie Persistierung, Fehlerredundanz, Robustheit usw. erlaubt es unterschiedlichste Anwendungssysteme mittels gleicher Basistechnologie zu realisieren.

Es wurde bei der Realisierung des TimeLess-Prototyps festgestellt, daß wegen einer fehlenden Universalarchitektur sehr viel Implementierungsredundanz entstanden ist. Eine auf die-

sem Architekturkonzept basierende Universalarchitektur hätte eine Realisierung und Implementierung wesentlich erleichtern können. Große Teile der verschiedenen Komponenten hätten anhand bereits vorhandener Beschreibungen anderer Komponententeile generiert werden können. Beispielsweise hätte sich die Cachelogik sowohl der IAEs, der LEs als auch der PEs aus der Verhaltensbeschreibung der anderen Komponententeile generieren lassen. Das PE als Attributverwalter ließe sich schätzungsweise zu 80% generieren. Dabei ist lediglich eine Auflistung und Klassifizierung aller Attribute notwendig. Diese läßt sich direkt aus der Verhaltensbeschreibung eines LEs ableiten.

Aber auch ohne Universalarchitektur konnte durch den realisierten TimeLess–Prototyp gezeigt werden, daß das hier vorgestellte Architekturmodell sehr viel Zukunftspotential besitzt. Die Entwicklung eines Frameworks, bzw. einer Universalarchitektur sollte eingehend geprüft und in Aussicht gestellt werden. Dadurch kann die praktische Umsetzung von Anwendungsanforderungen drastisch beschleunigt und vereinfacht werden. Das Ziel eines Anwendungsentwicklers ist schließlich nicht, sich mit Themen, wie Persistenzverwaltung, Verteilung oder Systementwurf zu beschäftigen. Ein vorhandenes Framework ließe die Konzentration auf die zwei Aufgabengebiete zu, die den Anwendungsentwickler interessieren: Die Geschäftslogik und die Präsentation durch die Benutzerschnittstelle. Nach den Erfahrungen des bisher realisierten Prototyps scheint diese Vision mit diesem Architekturkonzept möglich!

Anhang A Identifizieren und Zeigen

Bei der Entwicklung von programmierten Systemen kommt es oft zu Verständnisproblemen, weil der Unterschied zwischen einer Referenz und einem Identifikator nicht klar ist. Dies kann jedoch auf sehr natürliche Weise anhand von Gegebenheiten erläutert werden, die man auch in der Realität wiederfindet.

Um etwas identifizieren zu können, muß das zu identifizierende eine Identität besitzen. Insofern hat jeder Mensch eine eigene, eindeutige Identität (von jedem anderen Menschen unterscheidbar). Jedem Menschen können Attribute zugeordnet werden, die eine eindeutige Identifikation ermöglichen. Die Identität eines Menschen ist allerdings auch ohne Zuteilung von Attributen eindeutig. Attributierung zum Zwecke der Identifizierbarkeit ist allerdings notwendig, damit man überhaupt von oder über jemanden reden kann. So ist jeder Mensch durch seinen Namen, Geburtstag und seine Personalausweisnummer eindeutig identifizierbar. Jeder, der diese Attribute kennt, kann einen bestimmten Menschen identifizieren. Natürlich ist dies nur eine Möglichkeit, denn ein Lehrer wird seine Schüler nicht mit der Personalausweisnummer anreden, damit sie wissen, wer gemeint ist. In jeder Umgebung wird man eine hinreichend große Menge von Attributen finden, die zur eindeutigen Identifizierung aller Entitäten ausreicht. Ein Exemplar einer solchen Attributmenge kann man als Identifikator bezeichnen, denn es kann benutzt werden um jemandem mitzuteilen, wen man meint. Ein solcher Identifikator kann allerdings nur dann auch garantiert zur Identifizierung benutzt werden, wenn er unter allen Identifikatoren der zu identifizierenden Entitätenmenge eindeutig ist.

Als Beispiel sei eine Mutter genannt, die einer anderen Person den (für sie eindeutigen) Namen ihres Kindes mitteilt. Dieser für die Mutter als Identifikator fungierende Name muß für die andere Person nicht unbedingt als Identifikator ausreichen. Diese könnte nämlich in ihrem Bekanntenkreis noch jemanden kennen mit dem gleichen Namen. Aus diesem Grund wird Menschen in ihrer jeweiligen Umgebung ein eindeutiger Identifikator zugeordnet, der ein Unterscheidung von allen anderen Menschen in dieser Umgebung ermöglicht. Daher ist in einer globalen Umgebung die Einführung von Personalausweisnummern notwendig gewesen.

Solch ein eindeutiger Identifikator kann an andere weitergereicht werden, ohne daß diese die identifizierte Person überhaupt persönlich kennen müssen. Durch den Besitz eines Identifikators ist es möglich *über* einen Menschen zu reden. Es ist jedoch nicht unmittelbar möglich auch *mit* diesem Menschen reden zu können.

Will man hingegen mit jemandem reden, den man identifizieren kann, dann muß man in Erfahrung bringen, wo der Identifizierte sich befindet. Man muß auf ihn zeigen können. Dies kann man erreichen, indem man jemanden fragt, der bereits auf den Identifizierten zeigen kann, bzw. seinen Ort kennt. Dieser kann Auskunft über den Ort des Identifizierten geben.

Merkregel: Aus Zeigbarkeit folgt Identifizierbarkeit – aus Identifizierbarkeit folgt jedoch nicht Zeigbarkeit.

Um dies zu verdeutlichen, erfolgt an dieser Stelle ein Beispiel. Das Klassenzimmer einer Schule wird vom Schuldirektor besucht. Dieser kennt den Namen eines Schülers dieser Klasse und kann ihn somit identifizieren. Er kennt allerdings weder seinen Platz im Zimmer, noch wie er aussieht. Nach Anfrage beim Lehrer, der auf den betreffenden Schüler zeigt, kann der Direktor auch auf ihn zeigen und somit natürlich mit ihm reden, um ihm mitzuteilen, daß er sich umgehend im Sekretariat zu melden hat.

Eine zweite Möglichkeit diese Nachricht dem Schüler mitzuteilen, ist mit jemandem zu reden, der den Schüler kennt. So hätte der Direktor auch den Lehrer vor die Tür beten können, um ihm mitzuteilen, daß der Schüler sich sofort im Sekretariat zu melden hat. Die Mitteilung, um welchen Schüler es sich handelt, erhält der Lehrer vom Direktor indem er den als Identifikator fungierenden Namen des Schülers genannt bekommt.

Diese zweite Form ist eigentlich eine Mischform, denn es wird lediglich identifiziert, allerdings ausschließlich um zu zeigen und mitzuteilen. Dazu muß es jemanden geben, der sowohl identifizieren als auch zeigen kann. Dieser ist daraufhin in der Lage, anderen den Dienst anzubieten, Nachrichten an den Identifizierten weiterzuleiten, oder dem Fragenden den zeigbaren Aufenthaltsort des Identifizierten mitzuteilen.

A.1 Identifizieren und Zeigen – Technologie

In der Informationstechnik ist es notwendig, die beiden soeben beschriebenen Konzepte sauber zu trennen. Zur Identifikation wird meist eine minimale eindeutige Attributmenge herangezogen, die z.B. in der Welt der relationalen Datenbanken als Primärschlüssel bezeichnet wird. Ein Exemplar eines Primärschlüssels kann zur Identifikation einer Entität benutzt werden. Meist ist ein solcher Primärschlüssel ausschließlich zum Zweck der Identifikation eingeführt worden (z.B. eine laufende Zahl), und kann somit nicht als Eigenschaft der Entität bezeichnet werden.

Um auf eine Entität zeigen zu können, muß man eine sogenannte Referenz besitzen. Die Gefahr bei Referenzen liegt oft darin, daß sie nur auf einen bestimmten Ort zeigen (z.B. im Speicher), nicht jedoch zwingend auf die gemeinte Entität. Daher ist es wichtig, daß referenzierte Entitäten weder ihren Ort ändern, noch gelöscht werden.

In der Welt von CORBA kommen häufig verteilte Objekte vor, die man identifizieren kann, deren Ort jedoch ständig ein anderer sein kann. Daher besitzt CORBA einen Mechanismus,

der eine Referenzierung durch den Identifikator ermöglicht. Dabei kann man den Identifikator von CORBA in eine Referenz umwandeln lassen.

Solche CORBA–Referenz zeigen jedoch nicht direkt auf das identifizierte Objekt. Sie zeigen vielmehr auf ein Helferobjekt (Skeleton), welches das identifizierte Objekt kennt. Der Grund hierfür ist, daß eine CORBA Referenz plattformunabhängig sein muß, die eigentliche Objektreferenz ist jedoch plattformabhängig.

A.2 Identifikation und Referenzierung in TimeLess

Die bisher beschriebenen Zusammenhänge sollen nun an einem exemplarischen Aufbau einer TimeLess–Sitzung dargestellt werden. Das in Abbildung A–1 gezeigte Bild stellt eine Situation mit realistischen Zustandsbelegungen dar. Es zeigt die drei Ebenen der TimeLess Architektur (Client–Schicht, Logische Schicht, Persistenz–Schicht) sowie die Schicht der relationalen Datenbank, die bei TimeLess ebenfalls benutzt wird.

Man muß hier annehmen, daß die drei Regal–Elemente (IAE, LE und PE) auf den drei Schichten bereits existieren. Dabei ist das Regal–IAE „Physik“ in Besitz einer Referenz auf das Regal–LE „Physik“. Diese Referenz wird durch die Hilfe eines Stub–Skeleton Mechanismus von CORBA benutzt um Aufträge seitens des IAE an das LE zu schicken. Das „Physik“–LE wiederum ist in Besitz einer PE–ID[1]. PEs können allerdings – bedingt durch den Schlafen–Wecken–Mechanismus – ihren Ort zur Laufzeit ändern. Sie können an einem Art deaktiviert werden (Schlafen), um an einem anderen Ort wieder aktiviert zu werden (geweckt). Daher ist es wichtig, daß vor einer Referenzierung des PEs entweder die vorhandene Referenz überprüft wird, oder eine neue durch CORBA anhand der PE–ID ermittelt wird.

Im bisher beschriebenen Anfangszustand sind die Wertbelegungen der jeweiligen Listen leer. Das bedeutet, daß weder die IAE, noch die LE, noch die PE Brett Liste belegt sind. Ebenso existieren bisher noch keine Brett Elemente (weder IAE, LE, noch PE). Der nachfolgend beschriebene Vorgang zeigt welche Schritte notwendig sind, um auf der Client Schicht die drei Brettexemplare „Akustik“, „Optik“ und „Mechanik“ anlegen und initialisieren zu können.

Ein Benutzer möchte sich über den Inhalt des Regals informieren, also ein Liste der Bezeichnungen der in ihm enthaltenen Bretter erhalten. Dazu nutzt er einen Dienst des Regal IAEs. Dieser Aufruf wird wiederum über die vorhandene LE–Regal–Referenz an das LE weitergeleitet. Das LE kann den Auftrag noch nicht ausführen, da seine Brett–Liste noch unbelegt ist. Es wandelt daher seine PE–ID in eine PE–Referenz um, um vom Regal–PE 167 eine Liste aller Brett–IDs anzufordern. Diese Umwandlung wird durch den im Bild nicht dargestellten PE–Ort Verwalter erledigt, der im Architekturkapitel eingeführt wurde. Dies ist ein CORBA–orienterter Dienst, der in der Lage ist einen Identifikator in eine CORBA Referenz umzuwandeln.

[1] ID:= Identifikator

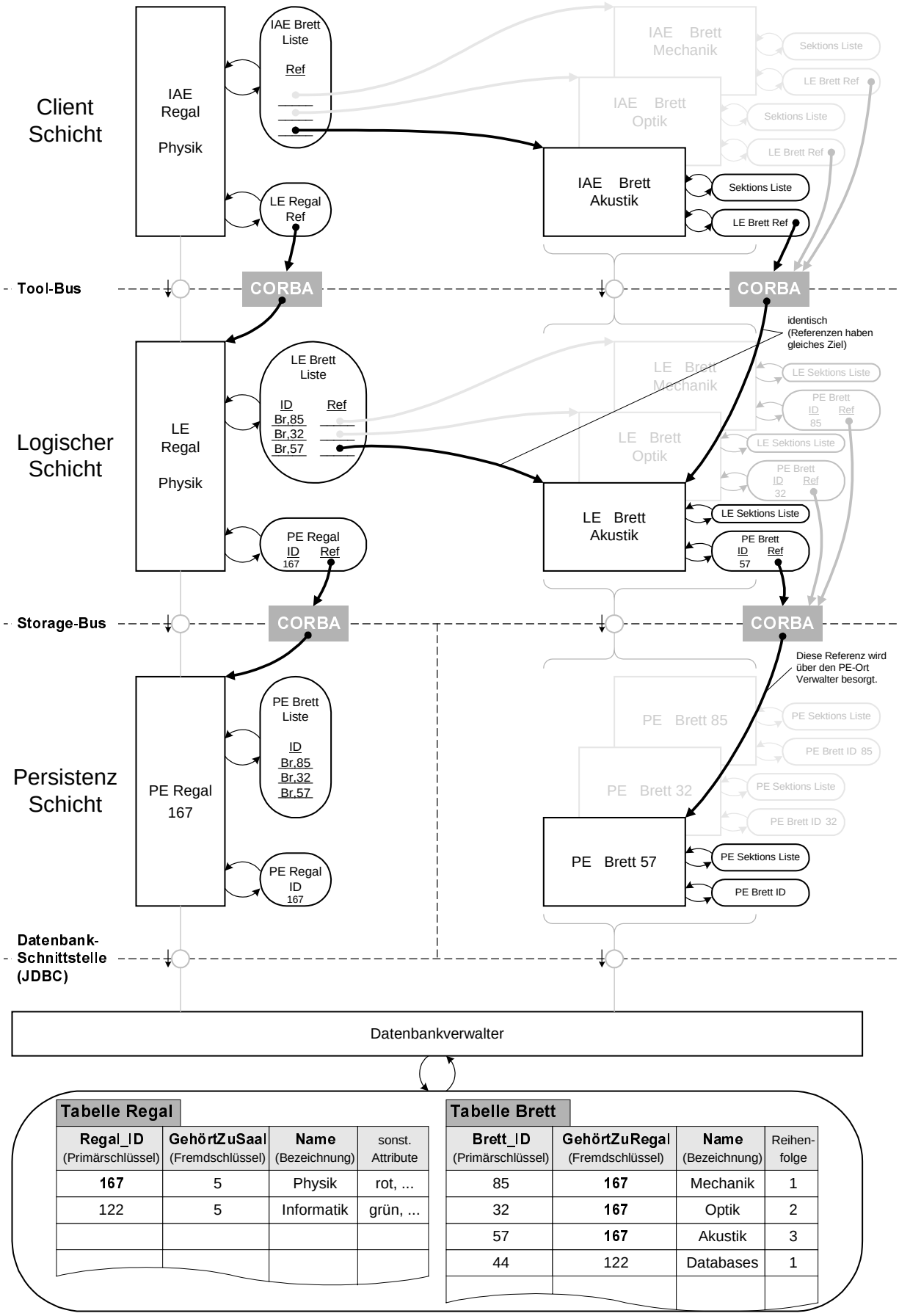

Abbildung A–1: Identifikation und Referenzierung von Objekten verschiedener Ebenen

Da das PE 167 aber auch noch nicht in Besitz der Liste der Brett IDs, muß es eine Datenbankanfrage starten, die als Ergebnis eine Liste aller Brett–IDs dieses Regals enthält. Diese Anfrage wird als SQL–Anfrage mittels des genormten Schnittstellenprotokolls JDBC[1] durchgeführt. Dieses deskriptive mengenorientierte Zugriffssprache erlaubt es, bestimmte Datenfelder der Datensätze zurückzuliefern, bei denen ein zu vergleichendes Prädikat mit dem Wert eines bestimmten Datenfeldes übereinstimmt. In diesem Fall sollen alle Brett–IDs der Tabelle „Brett“ zurückgegeben werden, deren Fremdschlüsselfeld „GehörtZuRegal“ mit dem Regal_ID–Wert „167“ übereinstimmt. Das Ergebnis der Anfrage ist eine Menge bestehend aus den drei Brett_IDs „85“, „32“ und „57“. Diese IDs trägt das Regal–PE in seine PE–Brett–Liste ein.

Den Inhalt dieser Brett Liste kann das PE nun dem anfragenden Regal–LE mitteilen. Dieses trägt die erhaltene Liste von Brett IDs ebenfalls in seine Brett Liste ein. Da die Client Schicht keine PE–IDs kennt, sondern nur CORBA–Referenzen auf LEs kennt, muß das Regal–LE zuerst die drei Brett–LEs erzeugen (lassen), um überhaupt in den Besitz der Referenzen auf die Brett–LEs zu kommen. Beim Erzeugen bekommen die Brett–LEs ihre zugehörigen PE–Brett–IDs mitgeteilt. Außer dem Wissen über die PE–ID ihres Partner–PEs ist der Zustand dieser Brett–LEs noch unbelegt.

Nun ist das Regal–LE in Besitz der Referenzen der neu erzeugten Brett–LEs. Diese Referenzen teilt es nun dem aufrufenden Regal–IAE mit. Dabei erfolgt von CORBA automatisch eine Erzeugung von Skeletons für die jeweiligen Brett–LEs und somit eine Wandlung der lokalen plattformabhängigen Referenzen der Logischen Schicht in eine allgemeine plattformunabhängige CORBA–Referenz. Dies ist für die Beteiligten LEs und IAEs jedoch transparent.

Das Regal–IAE muß nun ebenfalls drei Brett–IAEs erzeugen (lassen) und diesen neu erzeugten IAEs die Referenz ihrer zugehörigen Partner–LEs mitteilen. Dies ist notwendig, damit die Brett–IAEs Anfragen vom Benutzer an die entsprechenden LEs weiterleiten können. Sobald die drei Brett–IAEs erzeugt sind, trägt das Regal–IAE die lokalen clientabhängigen Referenzen der neu erzeugten Brett–IAEs in seine Brett–Liste ein.

Bemerkung: Diese lokalen Referenzen sind nicht die Referenzen der Brett–LEs. Das Regal–IAE kennt nur seine Brett–IAEs. Falls es Information über seine Bretter benötigt, stellt es die entsprechende Anfrage an die Brett–IAEs, nicht an die Brett–LEs.

Da der Benutzer eine Liste der Bezeichnungen der im Physik–Regal enthaltenen Bretter sehen möchte, muß das Regal–IAE nun alle Brett–IAEs nach ihrer Bezeichnung fragen. Diese kennen zu diesem Zeitpunkt jedoch nur ihre zugeordneten Partner–LEs. Daher leiten sie die Anfrage an die Brett–LEs weiter. Deren Zustand ist jedoch außer der ID ihrer Partner–PEs noch unbelegt. Daher müssen die PE–IDs in Referenzen umgewandelt werden. Die jeweiligen Brett–PEs sind jedoch noch nicht „wach“, d.h. sie existieren zwar als identifizierbare Objekte, müssen jedoch „zum Leben erweckt“ werden, damit sie Anfragen annehmen können.

Da die Umwandlung der PE–IDs in Referenzen vom PE–Ort Verwalter gemacht wird, ist dieser auch derjenige, der feststellt, ob ein PE zuerst geweckt werden muß. Ist dies der Fall, so wird das PE von einem festgelegten PE–Verwalter geweckt und initialisiert. Zu diesem In-

[1] JDBC:= Java DataBase Connectivity

itialisierungsvorgang gehört das Erfragen des PE–Zustands von der Datenbank. Nachdem dies geschehen ist, haben die so „geweckten“ PEs Kenntnis der von ihnen verwalteten Attributbelegungen. Der PE–Ort–Verwalter kann nun ebenfalls die angeforderten Brett–PE–Referenzen zurück an die Brett–LEs liefern. Diese sind nun mittels dieser Referenz in der Lage die jeweiligen PEs nach ihrer Bezeichnung zu fragen. Die so erhaltene Bezeichnung teilen sie den ursprünglich anfragenden Brett–IAEs mit. Diese sind jetzt in Besitz ihrer Bezeichnung, die sie wiederum dem fragenden Regal–IAE „Physik“ mitteilen.

Damit ist der komplette Vorgang einer Anfrage mit Erzeugung der entsprechenden Elemente abgeschlossen. Man erkennt die unterschiedlichen Semantiken, die bei der Identifikation und Referenzierung verschiedener Elemente unterschiedlicher Schichten in diesem System vorkommen können.

Anhang B Internet Technologie – ein kurzer Überblick

Eine der wichtigen Anforderungen an TimeLess ist dessen Nutzung in Umgebungen, in denen keine Administration oder Installation erforderlich ist. Große Unternehmen, deren Mitarbeiter weltweit operieren und ständig unterwegs sein können, sind in hohem Maße von ihren Informationssystemen abhängig. Damit diese Systeme den Mitarbeitern überall zur Verfügung stehen – unter Umständen auch in unternehmensfremden Rechnerumgebungen – müssen neue Technologien zum Einsatz kommen. Diese dürfen nicht mehr voraussetzen (wie bisher üblich) daß eine lokale Installation des jeweiligen Informationssystem erfolgt, bevor das System benutzt werden kann. Ein Ansatz zur Lösung dieser Anforderung wird durch die JAVA–Applet Technologie aufgezeigt, die im Rahmen des aktuellen TimeLess–Prototyps zum Einsatz gekommen ist. Ohne allzu sehr in die Tiefe dieser umfangreichen Technologie gehen zu wollen, sollen hier dennoch einige Auszüge davon präsentiert werden. Bei Bedarf existiert zu dieser Technologie sehr viel weiterführende Literatur. Ebenfalls kann auch weitere Informationen vom Autor und der TimeLess Entwicklungsmannschaft nachgereicht werden.

B.1 JAVA–Applet Technologie

JAVA–Applets, eine Bezeichnung die sich in etwa mit „kleine Anwendungen“ übersetzen läßt, haben durch den allgemeinen Aufschwung des Internet sehr schnell an Bedeutung gewonnen. Das Prinzip von Applets ist es, die Funktionalität eines WWW[1]–Browsers nachträglich und zur Laufzeit erweitern zu können. Ein WWW– oder Web–Browser ist eine für alle gängigen Plattformen vorhandene Internet–Anwendung, welche durch das Internet verfügbar gemachte Information für den Benutzer grafisch aufbereitet und präsentiert. Diese im Internet verfügbare Information wird in Form von HTML[2]–Dokumenten von WWW–Servern zur Verfügung gestellt. Diese WWW–Server können, sofern sie am internationalen Internet–Rechnerverbundsnetz angeschlossen sind, jedem der ebenfalls an diesem Netz angeschlossen ist,

[1] WWW:= World Wide Web, bzw. übersetzt „weltweites Netz“. Dieses Akronym steht für das als Internet bekannte Netz weltweit verfügbarer Informationsdienste.

[2] HTML:= Hypertext Markup Language, Eine Sprache, die sowohl Inhalt und Layout einer flächig statisch darstellbaren Seite definiert. Sie ist die Standardsprache der von Web–Browsern dargestellten Seiten.

Daten zur Verfügung stellen. Das wichtigste technische Protokoll, das bei solchen Datentransporten zum Einsatz kommt ist das HTTP–Protokoll[1]. Damit ein Web–Browser die entstehende Fülle von Information explizit identifizieren kann, wird ein weltweit eindeutiger URL–Identifikator[2] eingesetzt, der sowohl den Web–Server als Informationsanbieter identifiziert, als auch die von diesem Web–Server angebotene Seite als Träger von Information.

Um eine Seite eines Web–Servers anzuzeigen, muß man lokal einen WWW–Browser starten. Durch Angabe einer URL wird dem Benutzer durch den Browser der Inhalt der identifizierten HTML–Seite im Browserfenster dargestellt. Dabei wird die entsprechende HTML–Seite, die letztlich in Form eines ASCII–Texts geladen wird, vom Browser interpretiert. Dadurch können über die reine Textinformation hinaus zusätzlich noch als Text kodierte grafische Präsentationseigenschaften so dargestellt werden, daß der Seite in vielen Fällen gar nicht angesehen werden kann, daß es sich eigentlich um eine reine Textdatei handelt. Diese HTML–Seiten besitzen z.B. die Möglichkeit auf andere Formen von Daten zu referenzieren (auch wieder per URL). So können z.B. pixelorientierte Grafiken nachgeladen werden.

Die Mächtigkeit von HTML ist allerdings in vielen Dingen noch zu beschränkt. Es wurde schnell erkannt, daß es eine Möglichkeit geben müßte, die Funktionalität eines Browsers, über die HTML–Fähigkeiten hinaus, zu erweitern. Es mußte eine Möglichkeit geschaffen werden, die es ermöglicht nachladbare Programme vor Ort, also vom Browser, abwickeln zu können. Diese Programme sollten wie HTML–Seiten auch vom Browser nachgeladen werden können, um nach dem Ladevorgang als Interpretationsvorschrift für einen im Browser integrierten Abwickler zu dienen.

Der Abwickler, der dafür eingesetzt wird, ist die JAVA–Virtual Machine. Sie ist in der Lage JAVA–Bytecode[3] abzuwickeln. Dieser Bytecode kann jederzeit mit dem URL–Mechanismus nachgeladen werden. So ist es möglich, daß auch auf Rechnern, auf denen ausschließlich ein Browser installiert ist, fremde Programme ausgeführt werden können und damit die Funktionalität des Browsers beliebig erweitert werden kann. Diese anfangs sehr kleinen Programme (JAVA–Applets) haben durch den stetigen Reifeprozeß der Programmiersprache JAVA mittlerweile eine hohe Mächtigkeit erreicht. Zum Teil können damit nicht nur kleine Anwendungen (Applets) realisiert werden, sondern auch größere professionelle Anwendungen.

[1] HTTP:= Hyperttext Transfer Protocol, Das Standardprotokoll um WWW–Inhalte zu transportieren. Dieses Protokoll wird vor allen von Browsern benutzt um Daten von Web–Servern zu transportieren (z.B. HTML–Dateien)

[2] URL:= Uniform Resource Locator. Weltweit eindeutiger Identifikator zur Identifikation von Diensten, HTML–Seiten, Web–Server usw. im WWW.

[3] Java–Bytecode: Java Programmcode wird erst kompiliert in einen plattformunabhängigen Bytecode. Dieser Bytecode wird dann von Java Abwicklern interpretiert. Durch die Vielzahl der unterstützten Plattformen erlaubt dieser Mechanismus, Java Programme nur einmal zu schreiben, und auf allen Plattformen auszuführen. Ein aufwendige Portierung auf verschiedene Plattformen ist nicht notwendig.

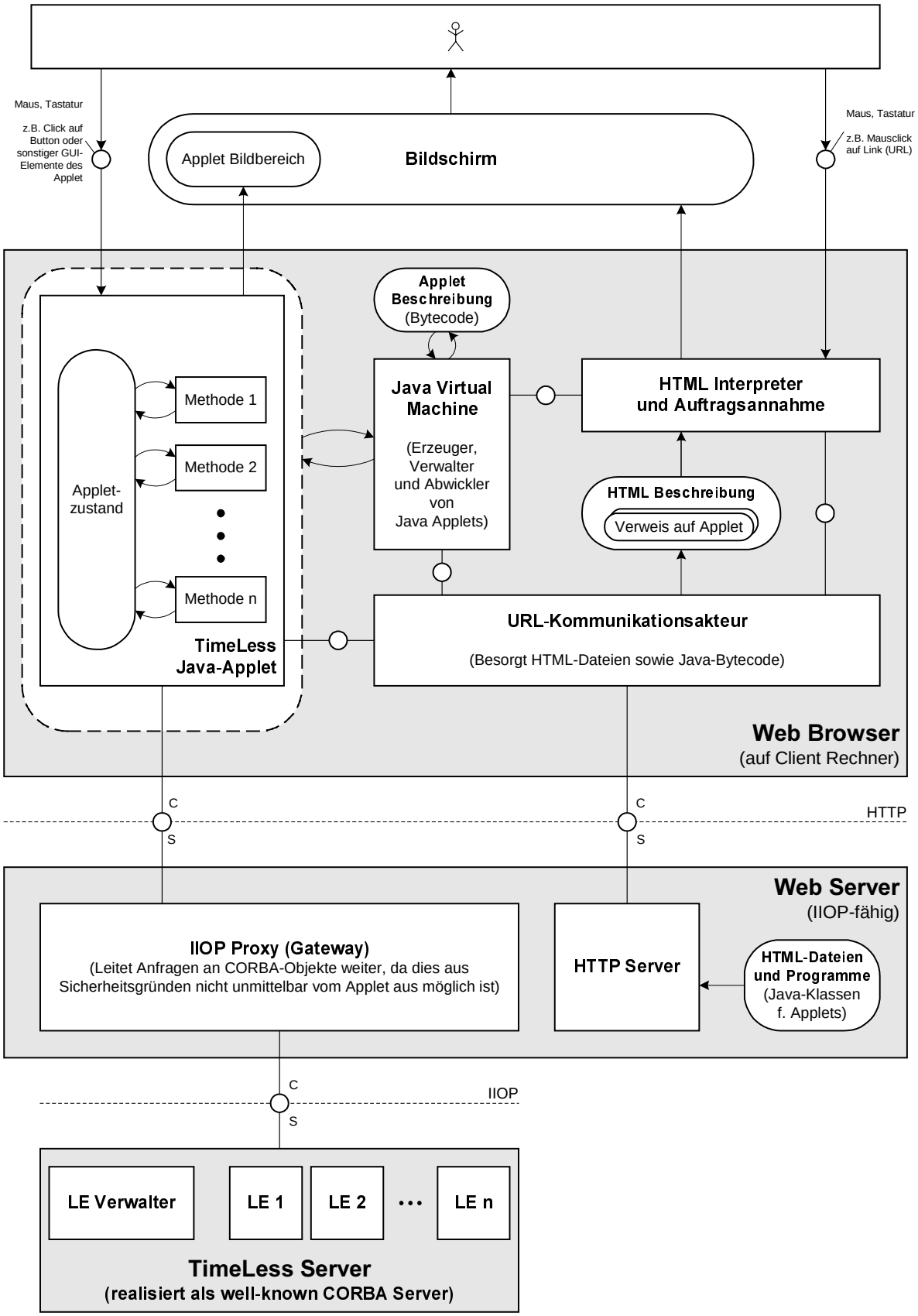

Abbildung B–1: Zero Installation mit JAVA Applet Technologie

Leider bringt dies jedoch auch große Sicherheitsprobleme mit sich, die es notwendig machen diese Applets in ihrer Funktionalität einzuschränken. Beispielsweise könnte ein bösartiges Applet die Festplatte des lokalen Hostrechners, auf dem es läuft, löschen. Durch die Offenheit des Internet ist dies nicht auszuschließen. Daher wurde nur unter bestimmten Umständen der Zugriff auf lokale Ressourcen erlaubt. Fremde, also im Internet verfügbare Ressourcen wurden grundsätzlich ausgeschlossen. Die einzigen Rechner, mit dem ein Applet kommunizieren darf, sind der lokale Rechner auf dem das Applet abgewickelt wird, und der Web–Server, von dem es selbst geladen wurde.

Will man nun global vernetzte Systeme bauen, die mittels Applet–Benutzerschnittstellen eine Interaktion mit dem Benutzer ermöglichen, dann muß man einen Weg finden diese Hürde zu überwinden, ohne dadurch das Sicherheitskonzept zu untergraben.

Trägersysteme zur Realisierung von Vermittlungs– und Transportdiensten, wie CORBA[1], hegen den Anspruch, weltweit verteilte Objekte miteinander kommunizieren zu lassen. So sind CORBA Server–Objekte von anderen CORBA Client–Objekten zugänglich, solange sie alle am gemeinsamen ORB[2] angemeldet sind.

Daher wurde CORBA als Vermittlungs- und Transportdienst bei dem während dieser Diplomarbeit realisierten TimeLess–Prototyp eingesetzt. Die angesprochenen Sicherheitsprobleme zusammen mit der JAVA–Applet Technologie konnten durch den Einsatz eines speziellen Produkts gelöst werden. Diverse Hersteller von CORBA–ORBs bieten hierfür unterschiedliche Möglichkeiten an.

Abbildung B–1 zeigt den Aufbau der Komponenten Web–Browser, Web–Server und TimeLess–Server. Man erkennt im Web Browser die Komponenten, von denen bisher die Rede war. Das neue an diesem Bild stellt der Web Server dar. Dieser besteht nicht nur aus HTTP Server, d.h. er kann Dateien über das HTTP–Protokoll anbieten, sondern er besitzt auch einen Gateway[3] in die CORBA Welt. Durch diesen Gateway ist es möglich, die klassische HTTP–Welt des Browsers zu verlassen und die IIOP[4]–Welt von CORBA zu betreten. IIOP ist das Protokoll, das benutzt wird um zwei verteilte Objekte mittels der in ihnen integrierten ORBs kommunizieren zu lassen.

Einem Applet ist es ja ermöglicht worden, Dienste des Web–Servers zu nutzen, von dem sein zugrundeliegender Bytecode herstammt. Ein Dienst dieses Web Servers besteht darin, IIOP–Proxy für JAVA Applets zu sein. Dieses IIOP–Proxy hat Zugang zu allen von diesem Rechner per Netzverbund angeschlossenen Rechnern. Durch das Internet bedeutet das, daß prinzipiell ein weltweiter Rechnerverbund verfügbar ist. Alle CORBA Server dieses Verbunds sind damit für den IIOP–Proxy verfügbar, und damit auch den JAVA–Applets, die

[1] CORBA:= Common Object Request Broker Architecture: Eine von der Object Management Group (OMG) in Zusammenarbeit mit vielen Instituten und Firmen definierter Objektverteilungsstandard. Dadurch wird ermöglicht, daß Objekte in riesigen Netzverbünden, miteinander kommunizieren können.
[2] ORB:= Object Request Broker. Eine zentrale Anmeldestelle, die dafür sorgt daß Anfragen an die entsprechenden Objekte weitergeleitet werden.
[3] Gateways werden benutzt, um einem System Zugang zu einem fremden System zu ermöglichen. Alle Komponenten des Systems müssen den zentralen Gateway benutzen um Dienste des anderen Systems nutzen zu können.
[4] IIOP:= Internet Inter–ORB Protocol; Das Standardprotokoll um die Kommunikation verschiedener ORBs zu ermöglichen Ein ORB ist eigentlich eine Komponente sowohl des Clients als auch des Rechners. Damit die ORBs dieser beiden Rechner miteinander kommunizieren können, wurde das IIOP spezifiziert.

mit für den IIOP–Proxy verfügbar, und damit auch den JAVA–Applets, die diesen Proxy nutzen.

Im Bild ist zu sehen, daß der TimeLess Server ein solcher CORBA Server darstellt. Dieser ist also durch diesen speziellen Web Server vom Applet aus zugänglich, ganz egal auf welchem Rechner dieser TimeLess Server läuft. Durch diesen Mechanismus ist im Prinzip das TimeLess Applet von jedem Browser aus nutzbar. Man muß lediglich die URL der TimeLess Homepage kennen.

Anhang C Darstellung von Kanälen

Im Rahmen dieser Diplomarbeit wurde eine Verfeinerung der syntaktischen Darstellung zweier Kanaltypen vorgenommen. Die entstandenen Darstellungen haben sich im Umfeld des hiesigen Software Engineering Labors als einfacher verständlich erwiesen. Darüber hinaus ist die genaue Semantik des jeweiligen Kanals besser erkennbar.

C.1 Teilnehmerkommunikationssystem zum Mithören

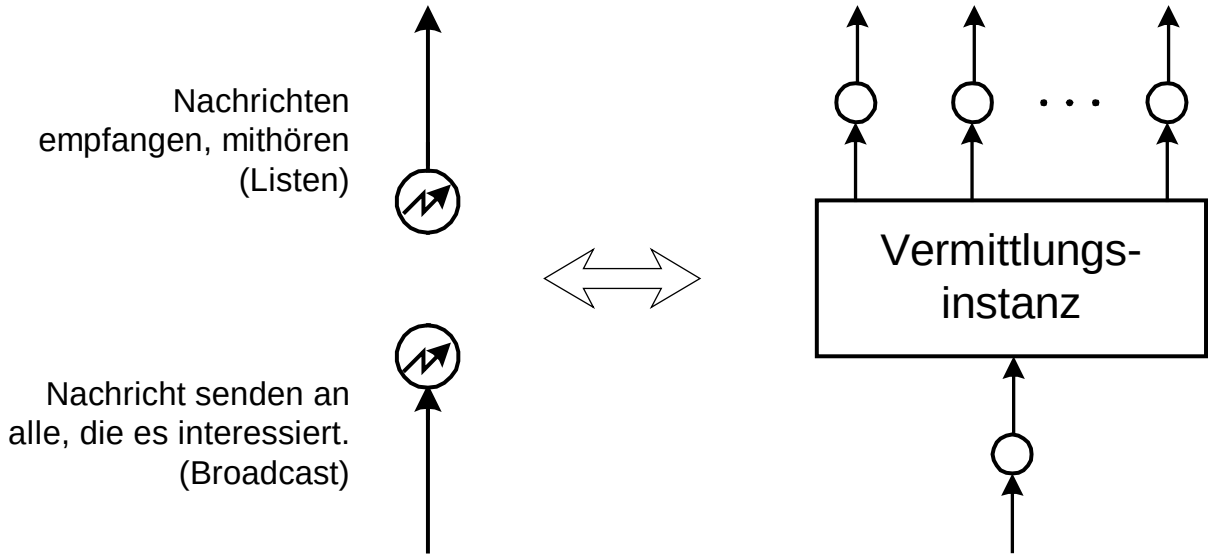

Abbildung C–1: Darstellungen eines Teilnehmerkommunikationssystems zum Mithören

Abbildung C–1 zeigt zwei Darstellungsmöglichkeiten eines Teilnehmerkommunikationssystems zum Mithören. Der linke Teil der Darstellung zeigt zwei Kanäle mit einem Blitz. Diese neue Darstellung steht für einen Kanal zwischen mehreren Teilnehmern eines Kommunikationssystems zum Mithören adreßfreier Botschaften. Diese Kanalsemantik wird oft als Broadcast/Listen–Semantik bezeichnet. Dabei ist ein Akteur Quelle einer Nachricht (Sender,

Broadcasters), ohne zu wissen ob oder wer diese Nachricht empfangen wird. Dies wird durch den Kanal symbolisiert, der aus Richtung des Senders zu einem „Blitzkanal“ führt.

Kommunikationspartner, die an Nachrichten interessiert sind, müssen für den Empfang von Botschaften bereit sein (Empfänger, Listener). Dies wird durch den Kanal mit Blitz in Richtung des Empfängers dargestellt. Empfänger kann es potentiell sehr viele geben, d.h. die Anzahl ist nur technisch beschränkt, nicht jedoch logisch.

C.2 Darstellung von Auftrags–/Rückmeldekanälen

Abbildung C–2 zeigt links die neue syntaktische Darstellung der bereits in anderer Form existierenden Darstellungen eines Auftrags–/Rückmeldekanals. Dabei ist der Kommunikationspartner an einem Ende des Kanals der anfragende Client (im Bild oben). Der Kommunikationspartner am anderen Ende des Kanals ist der rückmeldende bzw. antwortende Server (im Bild unten). Diese C/S–Darstellung erleichtert die Kommunikation insbesondere mit den Menschen, die mit der üblichen Aufbausyntax nicht sehr vertraut sind.

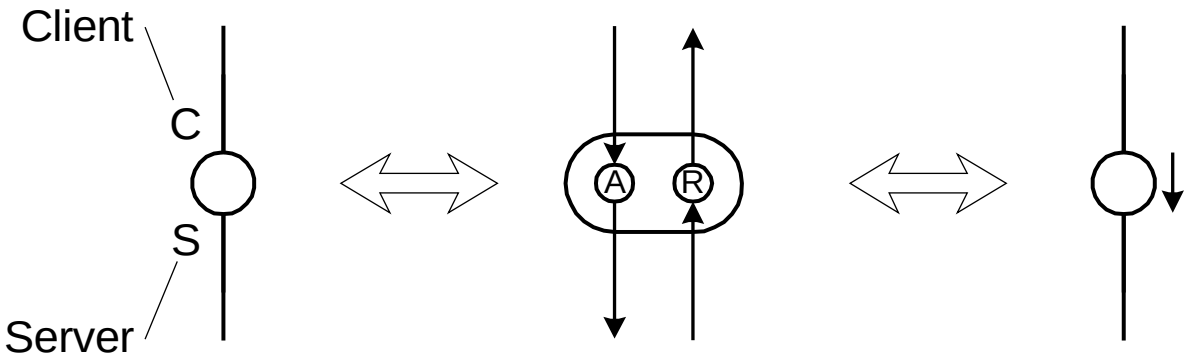

Abbildung C–2: Darstellungen eines Auftrags–/Rückmeldekanals

Literaturverzeichnis

[BMD94] Basis Modelling Department, SAP AG
Introduction to Concepts of the R/3 Basis System
Walldorf, Mai 1994

[BMD96] Basis Modelling Department, SAP AG
OMG Concepts–OMA and CORBA 2.0
Version 2.0
Walldorf, Februar 1996

[Brand97] Christian Brand
Spezifikation und Entwurf der Navigationskomponente für ein „Corporate Memory“ und Implementierung eines Demonstrationsprototyps
Diplomarbeit am Lehrstuhl für digitale Systeme,
Universität Kaiserslautern, 1997

[Bungert97] Andreas Bungert
Beschreibung programmierter Systeme mittels Hierarchien intuitiv verständlicher Modelle
Dissertation am Lehrstuhl für digitale Systeme,
Universität Kaiserslautern, 1997

[FZI97] Rainer Schmidt, Mechtild Wallrath
Gutachten zur Konzeption der 1. Phase des Projekts TimeLess
Forschungszentrum Informatik an der Universität Karlsruhe Karlsruhe, Mai 1997

[IAO97] Anette Weisbecker, Achristoph Altenhofen, Sven Bauer
Gutachten zur Konzeption von TimeLess aus Sicht des Dokumenten–Managements
Fraunhofer Institut f. Arbeitswirtschaft und Organisation, Stuttgart
Stuttgart, Oktober 1997

[Langhauser97] Jürgen Langhauser
Spezifikation und Entwurf des Schemas der Verwaltungsdaten eines „Corporate Memory“ und Implementierung der Zugriffskomponente
Diplomarbeit am Lehrstuhl für digitale Systeme,
Universität Kaiserslautern, 1997

[Orfali97a] Robert Orfali, Dan Harkey
Client/Server Programming with JAVA and CORBA
Wiley & Sons, 1997

[Orfali97b] Robert Orfali, Dan Harkey, Jerry Edwards
Instant CORBA
Wiley & Sons, 1997

[Royce70] W.W. Royce
Managing the development of large software systems: concepts and techniques
Proc. IEEE WESTCON, Los Angeles, 1970

[Sommerville96] Ian Sommerville
Software Engineering, 5. Auflage
Addison Wesley, 1996

[Visigenic97] Visigenic Software, Inc.
Visigenic VisiBroker Reference, User and Installation Manual
Vers. 3.0
Visigenic Software, 1997

[Vogel97] Andreas Vogel, Keith Duddy
Java Programming with CORBA
Wiley & Sons, 1997

[Wendt91] Siegfried Wendt
Nichtphysikalische Grundlagen der Informationstechnik
Springer, 1991