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

Diplomarbeit

Spezifikation und Entwurf des Schemas
der Verwaltungsdaten eines "Corporate Memory" und
Implementierung der Zugriffskomponente

Jürgen Langhauser
August 1997

Betreuer: Prof. Dr.-Ing. S. Wendt

Bearbeiter: Jürgen Langhauser
Vorholzstraße 12
76137 Karslruhe

Erklärung

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

Walldorf, den 30. August 1997

Jürgen Langhauser

Danksagung

An dieser Stelle möchte ich allen danken, die zum Gelingen dieser Arbeit beigetragen haben.

Mein Dank gilt in erster Linie Herrn Prof. Dr.-Ing. Siegfried Wendt, der die Diplomarbeit im Rahmen seiner Forschungsaktivitäten bei der Firma SAP AG ermöglichte und betreute. Sein mir entgegengebrachtes Vertrauen und die zahlreichen Gespräche, die zu einer sehr angenehmen Arbeitsatmosphäre beigetragen haben, waren außerordentlich hilfreich.

Meinen Kollegen Christian Brand, Markus Cherdron und Stefan Steißlinger, die mit mir das Projekt, in dessen Rahmen diese Arbeit entstanden ist, vorangebracht haben, danke ich für die leidenschaftlichen Diskussionen, die mir persönlich und fachlich viel geholfen haben.

Zuletzt möchte ich den Mitarbeitern des Lehrstuhls für Digitale System der Universität Kaiserslautern danken. Besonders zu Dank verpflichtet bin ich Bernhard Gröne, Eberhard Igelhaut, Andreas Knöpfel und Johannes Otto, die durch stete Hilfsbereitschaft und fachliche Kompetenz ihren Anteil an dieser Arbeit haben.

Inhaltsverzeichnis

Abbildungsverzeichnis

I Einleitung und Aufgabenstellung

Einleitung

Bei einem Softwareunternehmen als Produzent von Information entstehen zwei Arten von Information. Die eine wird an die Kunden ausgeliefert, die andere verbleibt in der Firma.

Die *ausgelieferte* Information ist die Software. Sie sichert durch ihren Verkauf den kurzfristigen Erfolg der Firma. Die im Unternehmen *verbleibende* Information ist das Wissen, das bei der Entwicklung und Produktion der Software entsteht. Durch dieses Wissen kann der langfristige Erfolg des Unternehmens gesichert werden, da bei der Neu- und Weiterentwicklung von Produkten eine bessere Qualität erreicht werden kann. Außerdem können bei der Produktion früher gemachte Fehler vermieden werden.

Das bei der Entwicklung und Produktion entstandene Wissen kann jedoch nur dann optimal genutzt werden, wenn eine gute Verteilung des Wissens auf die Köpfe der Mitarbeiter gewährleistet ist. Dazu muß der Zugang zur Information optimiert werden. Um dies zu erreichen, müssen die folgenden vier Bedingungen erfüllt sein:

1. Den Mitarbeitern sollte es zur Selbstverständlichkeit werden, ihr Wissen in Form von Dokumenten festzuhalten und abzulegen.

2. Dokumente sollten zentral verwaltet und abgelegt werden und für alle Mitarbeiter leicht zugänglich sein.

3. Die Mitarbeiter sollten in der Modellierung programmierter Systeme geschult werden.

4. Editoren zur Erstellung von Modellierungsdokumenten sollten vorhanden sein.

Jede der beschriebenen Bedingungen baut auf der vorhergehenden auf. Nur bei Erfüllung aller ist eine optimale Nutzung des Firmenwissens möglich. Die Erfüllung der ersten beiden Bedingungen soll sichern, daß die gesamte Information verfügbar ist. Die letzten beiden Bedingungen sollen erreichen, daß die Information in den Dokumenten optimal ihrer Verwendung entsprechend erstellt werden und somit die in ihnen enthaltene Information zu einem Wissenszuwachs des Lesers führt.

Das Projekt, in dessen Rahmen die vorliegende Diplomarbeit entstanden ist, befaßt sich mit der Erfüllung der zweiten Bedingung. Es soll ein System entwickelt werden, das die Lagerung und den Transport der Information eines Unternehmens unterstützt. Systeme dieser Art werden als ***Corporate Memory*** bezeichnet.

Der gewählte Ansatz ist die Erstellung eines Dokumentenverwaltungssystems. Durch dieses kann die Information mit den Dokumenten als Informationsträger gelagert und transportiert werden.

Aufgabenstellung

Das Dokumentenverwaltungssystem, mit dem die Benutzer kommunizieren, besteht, wie im Aufbaubild (Bild 1) dargestellt, aus der Dokumentenverwaltung und den von ihr benötigten persistenten, d.h. dauerhaft vorhandenen Daten.

Die Aufgaben, die im Rahmen dieser Arbeit bearbeitet wurden, betreffen die persistenten Daten. Eine Aufgabe bestand darin, benötigte Strukturen eines Teils der Daten zu entwickeln und zu spezifizieren. Eine weitere Aufgabe war die Ausarbeitung einer möglichst offenen und leistungsfähigen Architektur zur permanenten Haltung dieser Daten.

Die erarbeiteten Lösungen zur Strukturierung der Daten werden in Kapitel II vorgestellt. Die Betrachtungen zur Architektur werden in Kapitel III beschrieben.

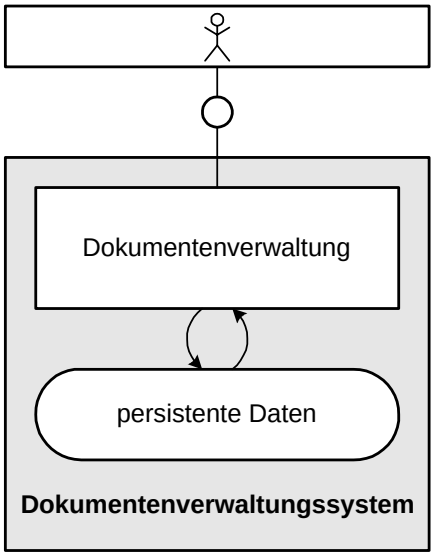


Bild 1: Aufbaubild eines Dokumentenverwaltungssystems

II Struktur der Verwaltungsdaten

Die Daten, deren Struktur im Rahmen dieser Arbeit bestimmt werden soll, sind die sogenannten ***Verwaltungsdaten***. Die Verwaltungsdaten umfassen alle Daten, die spezifisch für die Dokumentenverwaltung sind. Daten für ein Berechtigungswesen sind deshalb nicht Thema dieser Arbeit. Es ist nämlich ohne weiteres möglich, daß die Strukturen der Berechtigungsdaten beispielsweise auch für ein Finanzbuchhaltungssystem nutzbar sind.

Um die Strukturen der Verwaltungsdaten verstehen zu können, ist ein Einblick in die grundlegenden Konzepte der Dokumentenverwaltung nötig. Aus diesem Grund ist das Kapitel auch eine Beschreibung grundlegender Konzepte der Dokumentenverwaltung.

Die Beschreibung der Struktur erfolgt durch Entity-Relationship-Diagramme (ER-Diagramme). Die Attribute der Entitäten sind jeweils am Ende eines Unterpunkts aufgelistet. Da einige Attribute der Entitäten stark technisch bedingt sind, ist die Vollständigkeit nicht gewährleistet. Die Arbeit ist nur eine Vorüberlegung, weshalb einige Attribute, die aufgrund von Implementierungszwängen hinzukommen, fehlen.

Grundprinzip des Entwurfs von Datenstrukturen ist die Erkenntnis, daß sie schwerer zu ändern sind als Algorithmen. Aus diesem Grund ist die Struktur der Verwaltungsdaten offener gewählt als es für die Anwendung nötig gewesen wäre.

In Abschnitt eins werden die Dokumente und Bausteine erläutert, die als Informationsträger aufbewahrter Information dienen. Abschnitt zwei beschreibt die Möglichkeit, diese zu versionieren. Die zur Aufbewahrung der Dokumente bzw. Bausteine nötigen Strukturen und Prozesse werden in Abschnitt drei über Ablage, Transportwesen und Container behandelt. Möglichkeiten der Kommentierung im Dokumentenverwaltungssystem zeigt Abschnittt vier, Referenzen behandelt Abschnitt fünf.

1. Dokumente und Bausteine

Dokumente sind Informationsträger, die im Dokumentenverwaltungssystem zur Aufbewahrung von Information abgelegt werden. Nicht nur alles auf Papier Druckbare wie eine Diplomarbeit oder ein Vertrag, sondern auch Multimediaelemente können aufbewahrt werden. Das bedeutet, daß Video- und Tondokumente genauso wie einfache Textdokumente gesammelt werden. Den größten Teil der gesammelten Dokumente eines Softwareunternehmens machen die druckbaren Dokumente aus. Diese werden im folgenden als *flächig-statische Dokumente* bezeichnet.

Flächig-statische Dokumente sind oft Kombinationen aus mehreren elementaren Bausteinen wie Texte, Bilder oder Grafiken. Will man Teile eines solchen Dokuments in anderen verwenden, ohne daß die dort enthaltene Information mehrfach abgespeichert werden muß, ist es notwendig, daß die Struktur des Dokuments dem System bekannt ist. Nur dann ist es möglich, einzelne Bausteine abzuspeichern und mit Hilfe der Dokumentstrukturen verschiedene Dokumente als Kombination der Bausteine zu generieren.

Eine Verwaltung der Struktur ist jedoch nur dann möglich, wenn das Dokument ein systemeigenes Format besitzt. Verfügt es über kein systemeigenes Format, was durch die hohe Integration verschiedenster Editoren sehr wohl möglich sein kann, dann kann das Dokument nur als elementarer Baustein für andere Dokumente verwendet werden.

Aus dieser Beschreibung läßt sich im folgenden das in Bild 2 dargestellte Klassifikationsschema ableiten.

Ein Dokument gehört aufgrund seines Formats entweder der Klasse der ***Standard-Dokumente*** oder der Klasse der ***Non-Standard-Dokumente*** an.

Ist es den Standard-Dokumenten zuzuordnen, liegt es in Form des systemeigenen Formats vor. Ein Standard-Dokument besitzt eine durch das System verwaltete Struktur und kann Referenzen auf andere Dokumente enthalten, die durch das System interpretierbar sind. Liegt ein Dokument nicht in einem systemeigenen Format vor, gehört es der Klasse der Non-Standard-Dokumente an, deren innere Struktur dem System unbekannt ist.

Standard-Dokumente sind im Gegensatz zu den Non-Standard-Dokumenten immer statisch-flächig und somit druckbar. Ist ein Non-Standard-Dokument als statisch-flächig erkennbar, ist es der Klasse der ***bausteintauglichen*** Dokumente zuzuschreiben, d.h. sie können als elementare Bausteine in Standard-Dokumenten verwendet werden. Dokumente, die nicht statisch-flächig sind, gehören der Klasse der ***bausteinuntauglichen*** Dokumente an. Video- oder Audiodokumente sind bausteinuntauglich, da ein Videodokument nicht statisch und ein Audiodokument weder statisch noch flächig ist.

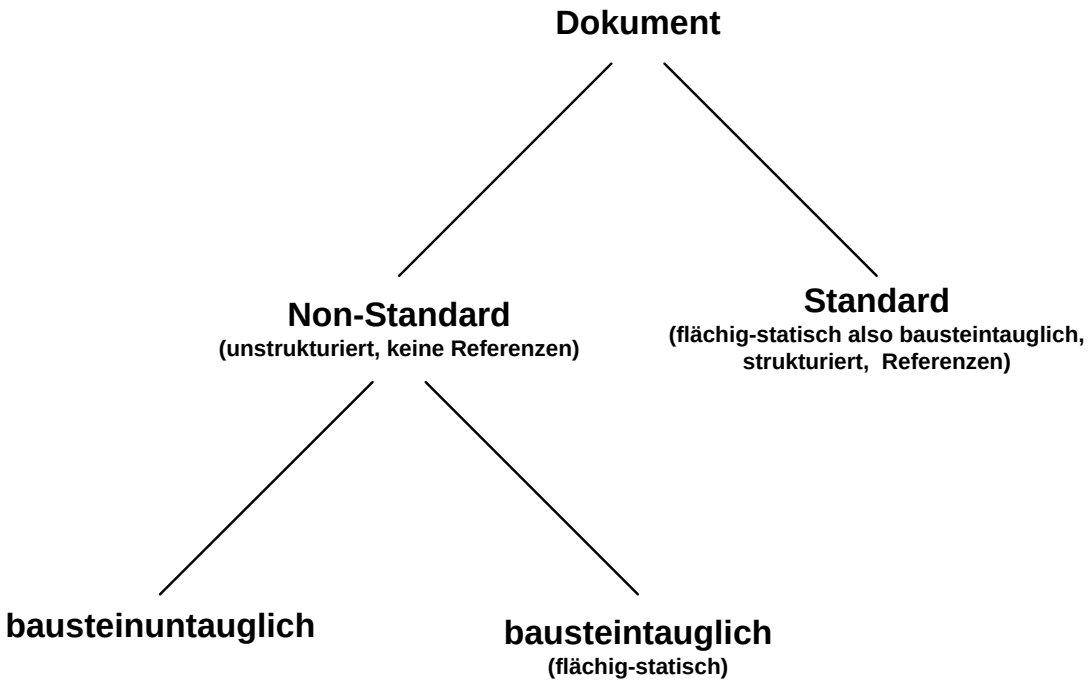


Bild 2: Klassifikationsbaum eines Dokuments

1.1 Strukturierung von (Standard) Dokumenten

Mit einem Standard-Dokument kann man die Vorstellung eines beklebten Blatt Papiers verbinden, wie es in Bild 3 angedeutet ist. Anstatt der grauen Flächen kann man sich einen Text, eine Grafik oder alles auf einem Papier Darstellbare vorstellen. Das Beispiel zeigt ein Papierstück, das mit zwei anderen Papierstücken beklebt ist. Eines der aufgeklebten Stücke ist wiederum mit einem anderen behaftet. Die Papierstücke sollen im folgenden als ***Bausteine*** des gesamten Dokuments bezeichnet werden. Ein Baustein besteht aus der beklebten Fläche, die ***Komponententdummy*** genannt werden soll, sowie aus der für den Betrachter später sichtbaren Fläche, die man als ***Bausteinbasis*** bezeichnet.

Betrachtet man das Beispiel näher, erkennt man, daß das Dokument baumstrukturiert ist. Jeder Knoten entspricht einem Baustein. Das unterste Papierstück ist der Wurzelknoten oder auch der Wurzelbaustein. Die obersten Papierstücke sind die Blätter.

Um es möglich zu machen, daß der Autor eines solchen Dokuments bei der Erstellung auf andere bereits strukturierte Dokumente zurückgreifen kann, ohne deren Struktur kennen zu müssen, ist dieses rekursiv definiert. Das bedeutet, daß ein Dokument genau aus einem Baustein und einer möglicherweise leeren Menge von Dokumenten besteht. Für das Beispiel heißt das, daß das Papierstück, das im Bild die Nummer 1 trägt, mit zwei Dokumenten beklebt ist. Das eine Dokument besteht dabei nur aus einem Baustein, das andere aus einem Baustein und einem Dokument.

Ist ein Dokument Teil eines anderen Dokuments, bezeichnet man es als ***Komponente.*** Die freien Flächen eines Baustein, die mit Komponenten beklebt werden, nennt man Komponententendummies, da diese Stellvertreter der Komponenten in einem Baustein sind.

Vorteil dieser Art des Erstellens von Dokumenten ist, daß man aus der Strukturierung eines Dokuments heraus die Mehrfachverwendung von Dokumenten unterstützt. Man geht automatisch dazu über, Kleinstdokumente zu erstellen, um sie dann zu größeren zu kombinieren. Dies ist die Art, wie normalerweise bei der Erstellung größerer Dokumente vorgegangen wird. Es ist nicht natürlich, ein Buch von Anfang bis Ende zu schreiben. Man schreibt normalerweise Kapitel für Kapitel und kombiniert diese im Nachhinein miteinander.

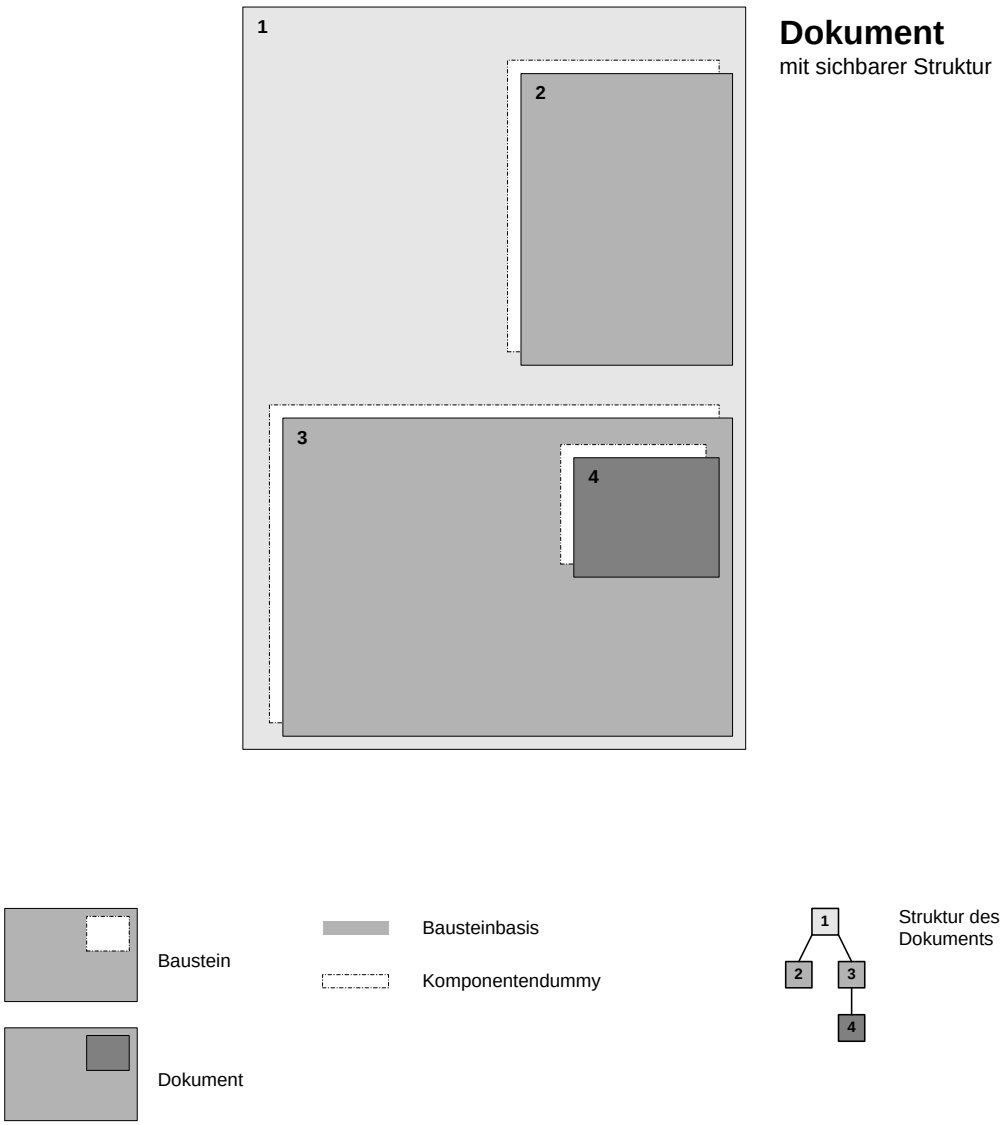


Bild 3: Beispiel eines Standard-Dokuments

Außer der Möglichkeit, Dokumente mehrfach zu verwenden, sollen Autoren auch für die Mehrfachverwendung ihrer Bausteine Unterstützung erhalten. Dies ist sinnvoll, um standardisierte Dokumente durch Vorlagen schaffen zu können. Beispielsweise könnte ein Autor einen Baustein erstellen, der das Erscheinungsbild eines Geschäftsbriefes standardisiert. Mit unterschiedlicher Belegung der im Baustein enthaltenen Komponentendummies entstehen dann verschiedene Briefe.

Um dies zu unterstützen, ist jeder Baustein, dessen Komponentendummies belegt sein können, Exemplar eines Bausteintyps. Das bedeutet, daß jeder neu erstellte Baustein zuerst als Bausteintyp abgespeichert wird. Wenn der Baustein Teil eines Dokuments wird, so ist dieser ein Exemplar des nun bestehenden Bausteintyps. Durch diese Verfahrensweise kann man erreichen, daß Bausteine in Form von Bausteinexemplaren durch unterschiedliche Belegung ihrer Komponentendummies zu unterschiedlichen Dokumenten führen.

Abschließend zur Diskussion von strukturierten Elementen ist zu bemerken, daß es außer der im Beispiel beschriebenen Art, Komponentendummies zu belegen, noch zwei weitere Belegungsarten gibt. Die bereits besprochene Art nennt man ***layoutvoraussetzend***. Ebenso können Dummies noch ***layoutgenerierend*** bzw. als ***Anlage*** belegt werden.

- *Layoutvoraussetzend*

  Eine Belegungsart, die eine vorgegebene Fläche im Baustein voraussetzt, um diese durch eine Komponente zu ersetzen, nennt man layoutvoraussetzend. Bei solch einer Belegung ist das Layout des späteren Dokuments durch dessen Wurzelbaustein bereits festgelegt.

- *Layoutgenerierend*

  Bei der layoutgenerierenden Belegung ist das Layout des Dokuments durch dessen Wurzelbaustein nicht festzulegen. Man kann die Fläche einer noch nicht bekannten Textkomponente nicht im voraus bestimmen. Der Text könnte beispielsweise zwei Zeilen lang sein, aber auch fünf Seiten beanspruchen. Bei der layoutgenerierenden Belegung würde eine solche Textkomponente zu einem neuen Layout führen. Unter Umständen erfolgen neue Seitenumbrüche.

- *Anlage*

  Bei der Belegung eines Komponentendummies als Anlage wird die Komponente an den Baustein angehängt. Bei einer Bewerbung werden beispielsweise dem Anschreiben noch Zeugnisse, Lebenslauf usw. zugefügt. Das eigentliche Anschreiben wäre dann der Wurzelbaustein des Dokuments, und Zeugnisse, Lebenslauf etc. wären Komponenten, die Komponentendummies des Wurzelbausteins als Anhang belegen.

Datenstruktur

Aus diesen Überlegungen folgt das in Bild 4 abgebildete ER-Diagramm der Struktur der Dokumente und Bausteine.

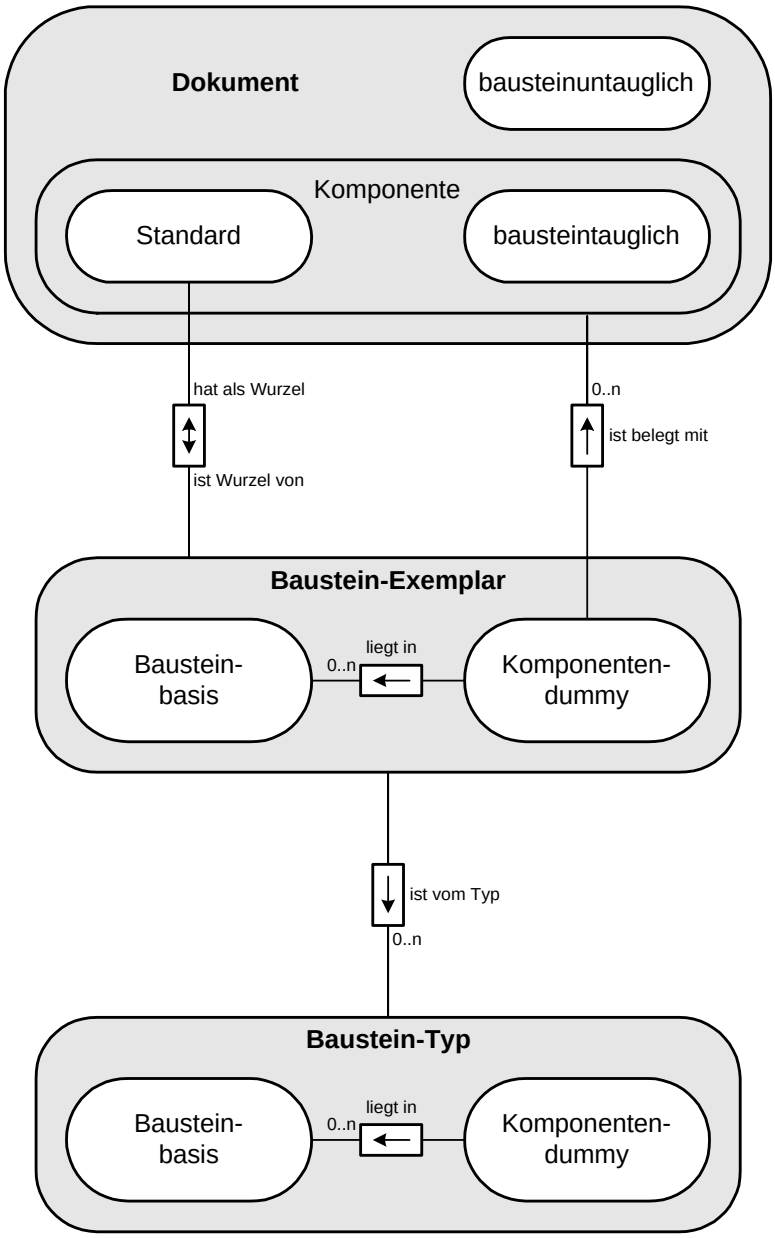


Bild 4: ER-Diagramm der Dokumentenstruktur

Das Bild teilt sich in drei Bereiche. Im oberen Bereich spricht man von Entitäten des Typs Dokument, im mittleren von Bausteinexemplaren und im unteren von Bausteintypen. Bausteine sind strukturierte Entitäten, sie bestehen aus einer Bausteinbasis und einer möglicherweise leeren Menge von Komponentendummies, die in der Bausteinbasis liegen. Jedes Bausteinexemplar ist Exemplar eines Bausteintyps. Von jedem Bausteintyp kann es beliebig viele Exemplare geben. Die Dokumente sind entweder Entitäten des Typs Standard-Dokument

oder Non-Standard, wobei diese in bausteintaugliche und bausteinuntaugliche Dokumente unterschieden werden. Jedes Standard-Dokument besteht mindestens aus einem Baustein, der Wurzel des strukturierten Dokuments ist. Jedes Bausteinexemplar ist Wurzel genau eines Dokuments. Die Komponentendummies eines Bausteinexemplars können dabei durch Komponenten belegt werden. Als Komponenten sind immer nur statisch-flächige Dokumente zugelassen, d.h. nur Standard-Dokumente und bausteintaugliche Non-Standard-Dokumente können Komponenten sein. Die Komponentendummies müssen nicht mit Dokumenten belegt sein. Ist ein Komponentendummy eines Bausteinexemplars nicht belegt, so ist jedes Dokument, das diesen Baustein enthält, ein unfertiges Dokument.

1.2 Klassifikation der Dokumente

Außer der Klassifikation eines Dokuments aufgrund seines Formats lassen sich noch drei weitere Einteilungen festlegen.

- *Wegweiser- / Sammlungsdokument*

  Wegweiserdokumente sind Dokumente, die Informationen für die Benutzer der Dokumentablage enthalten. Durch diese erhält der Benutzer Zusatzinformationen über die Ablage, die ihm die Wegfindung erleichtern sollen. Sie können durch das System interpretierbare Referenzen auf Elemente der Ablage enthalten. Deshalb liegen sie auch nur in Form von Standard-Dokumenten vor.

  Als Sammlungsdokumente bezeichnet man die eigentlich zur Aufbewahrung bestimmten Dokumente. Sammlungsdokumente können Dokumente jeglichen Formats sein.

- *Fertiges / Unfertiges Dokument*

  Ist ein Dokument unfertig, so darf es verändert werden; gehört es zur Klasse der fertigen Dokumente, darf deren Inhalt nicht mehr verändert werden. Fertige Dokumente können nur noch versioniert werden.

  Die Einteilung in unfertige und fertige Dokumente beruht darauf, daß Dokumente im systemeigenen Format während der Strukturierungsphase nur teilweise belegte Komponentendummies enthalten können. Solche Dokumente dürfen nicht öffentlich gemacht werden.

- *private / öffentliche (private / shared) Dokumente*

  Ein privates Dokument ist nicht zur Benutzung durch andere Autoren freigegeben, d.h. ein Autor, der nicht Ersteller dieses Dokuments ist, darf weder darauf referenzieren noch es als Komponente für seine eigenen Dokumente verwenden. Um eine Inkonsistenz zu vermeiden, sind deshalb alle Dokumente, die verändert werden können, also unfertig sind, immer private Dokumente. Damit ist gewährleistet, daß ein Autor ein Dokument verwenden kann, ohne daß es ein anderer Autor verändert. Ist ein Dokument öffentlich, kann es von anderen Autoren gelesen werden und als Komponente benutzt werden.

1.3 Attribute

Nachdem in den Abschnitten 1.1-1.3 alle Entitätstypen und Relationen im Bereich der Dokumente und Bausteine eingeführt wurden, sollen nun die Attribute der Entitäten besprochen werden.

In den Bildern 5, 6, 7 und 8 sind alle Entitäten mit ihren Attributen dargestellt. Die Darstellungsform ist so gewählt, daß alle (abgerundeten) Rechtecke entweder Entitätsklassen oder Entitäten darstellen. Attribute einer Entitätsklasse gelten für alle Entitäten dieser Klasse. In Bild 5 bedeutet das, daß die Dokumentattribute für alle Dokumententitäten gelten. Zusätzlich haben alle Entitäten, die der Klasse der Non-Standard-Dokumente angehören, die Attribute, die dort aufgelistet sind. Zur besseren Überschaubarkeit sind die Dokumente in Bereiche eingeteilt. Die Einteilung ist in den Darstellungen durch die Gruppierung der Attribute angedeutet.

1.3.1 Dokumente

Bild 5: Attribute der Dokumente

Identifikation

- **Name**

  Zur Identifikation durch den Benutzer besitzt das Dokument einen Namen, der außer zur Identifikation auch etwas über die Semantik des Dokuments aussagen soll, da Namen wie “LA_500” sind wenig aussagekräftig sind.

Erstellung

- **Ersteller:**

  Autor des Dokuments.
- **Erstellungsdatum/uhrzeit:**

  Datum und Uhrzeit der Erstellung des Dokuments.

Änderung

- **I_Änderungsdatum/uhrzeit:**

  Datum und Uhrzeit, zu der die Information, deren Träger das Dokument ist, zuletzt geändert wurde.
- **V_Änderer:**

  Person, die die Verwaltungsinformation eines Dokuments zuletzt geändert hat.
- **V_Änderungsdatum/uhrzeit:**

  Datum und Uhrzeit, zu der die Verwaltungsinformation zuletzt geändert wurde.

Beschreibung

- **Beschreibungstext:**

  Bei großen Dokumenten ist ein kurzer Textsinnvoll. der den Inhalt des Dokuments beschreibt.
- **Schlüsselwörter (Menge):**

  Zur Unterstützung der Suche nach Dokumenten kann ein Dokument Menge sogenannte Schlüsselwörter (Deskriptoren) besitzen. Durch diese wird der Inhalt kategorisiert.

Dokumentgröße

- **Informationserfassungsaufwand:**

  Der Informationserfassungsaufwand ist der Aufwand (die Zeitspanne), die ein Benutzer durchschnittlich braucht, um die in einem Dokument enthaltene Information zu erfassen. Der Informationerfassungsaufwand ist die einzige gemeinsame Größe, mit der man den Inhalt eines Dokuments messen kann, da die ablegbaren Dokumente zu unterschiedlich sind.

Klassifikation

- **Bereich:**

  Der Bereich gibt an, ob ein Dokument ein öffentliches oder privates Dokument ist.
- **Zustand:**

  Aktueller Zustand eines Dokuments. Ein Dokument kann entweder im Zustand fertig oder unfertig sein.

Berechtigung

- **Äquivalenzklasse:**
  Über die Äquivalenzklasse, der das Dokument angehört, kann man unter Zuhilfenahme des Berechtigungswesens die Zugriffsrechte auf das Dokument ableiten.

Performance

- **mbc (meantime between change):**
  Durchschnittliche zu erwartende Zeitspanne zwischen Änderungen eines Elements. Diese Information kann genutzt werden, um Caching-Mechanismen zu optimieren.

Darstellung

- **View:**
  Angabe über die Darstellungsart des Dokuments innerhalb der Ablage

Versionierung

- **Versionsnummer:**
  Ordnungszahl eines Dokuments innerhalb einer Versionskette.

Typ

- **Mimetype:**
  Format eines Dokuments. Konform mit der im Internet üblichen Formatangabe.

Verwaltete Information

- **Basisinformation:**
  Freilängenattribut das die durch das Dokument verwaltete Information beinhaltet. Dadurch, daß die Größe des Attributs nicht bestimmt ist, können beliebig große Informationsmengen durch das Attribut beinhaltet werden. Man nennt solche Attribute auch BLOBs (Binary Large Objekt).

1.3.2 Baustein

Bild 6: Attribute der Bausteine

Identifikation

- **Name**
  Name des Bausteintyps.

Erstellung

- **Ersteller:**
  Autor des Bausteintyps.
- **Erstellungsdatum/uhrzeit:**
  Datum und Uhrzeit der Erstellung des Bausteintyps.

Änderung

- **Änderungsdatum/uhrzeit:**
  Datum und Uhrzeit, zu der die Information, deren Träger der Bausteintyp ist, zuletzt geändert wurde.
- **V_Änderer:**
  Person, die die Verwaltungsinformation des Bausteintyps zuletzt geändert hat.
- **V_Änderungsdatum/uhrzeit:**
  Datum und Uhrzeit, zu der die Verwaltungsinformation zuletzt geändert wurde**.**

Beschreibung

- **Schlüsselwörter (Menge):**
  Analog zu Dokument.

Bausteingröße

- **Informationserfassungsaufwand:**
  Analog zu Dokument.

Klassifikation

- **Bereich:**
  Analog zu Dokument.

Berechtigung

- **Äquivalenzklasse:**
  Analog zu Dokument.

Performance

- **mbc (meantime between change):**
  Analog zu Dokument.

1.3.3 Bausteinbasis

Bild 7: Attribute der Bausteinbais

Verwaltete Information

- **Basisinformation:**
  Analog zu Dokument

1.3.4 Komponentendummy

Bild 8: Attribute der Komponentendummy

Belegung

- **Belegungsart:**
  Art wie ein Komponentendummy belegt werden kann. Beschreibung siehe auch Seite 7.

2. Versionierung

Dokumente können ebenso wie Bausteine versioniert werden. Eine Versionierung ist nur in Form einer Versionskette möglich, d.h. ein Element kann nur eine Nachfolgeversion und eine Vorgängerversion besitzen.

Aufgrund der Strukturierungsmöglichkeiten der Ablage ist es auch nicht nötig, eine weitere Versionsverwaltung im System vorzusehen. Ein Mechanismus, der das sogenannte Aus- bzw. Einchecken von Dokumenten ermöglicht, um eine Konkurrenzsituation bei der Versionierung zu vermeiden, ist nicht vorgesehen. Bei einer klaren Einteilung, wer ein Dokument versionieren darf, und einer guten Arbeitsorganisation innerhalb eines Unternehmens, ist normalerweise kein Bedarf für einen solchen Mechanismus.

In Bild 9 ist das zur Versionstruktur gehörige ER-Diagramm dargestellt. Als versionierbare Enitäten erkennt man Non-Standard-Dokumente, Bausteine und die Standard-Dokumente, die als Sammlungsdokumente in das System eingebracht werden. Wegweiserdokumente sind nicht versionierbar, da zwischen den einzelnen Wegweiserdokumenten keine Versionsbeziehung existieren kann. Beispielsweise ist der Bauplan eines abgerissenen Hauses kein Vorgänger des Bauplans des neuen Hauses. Genausowenig ist ein Wegweiserdokument für eine nicht mehr existierende Ablagestruktur keine Vorgängerversion für ein Wegweiserdokument der darauf folgenden Ablagestruktur.

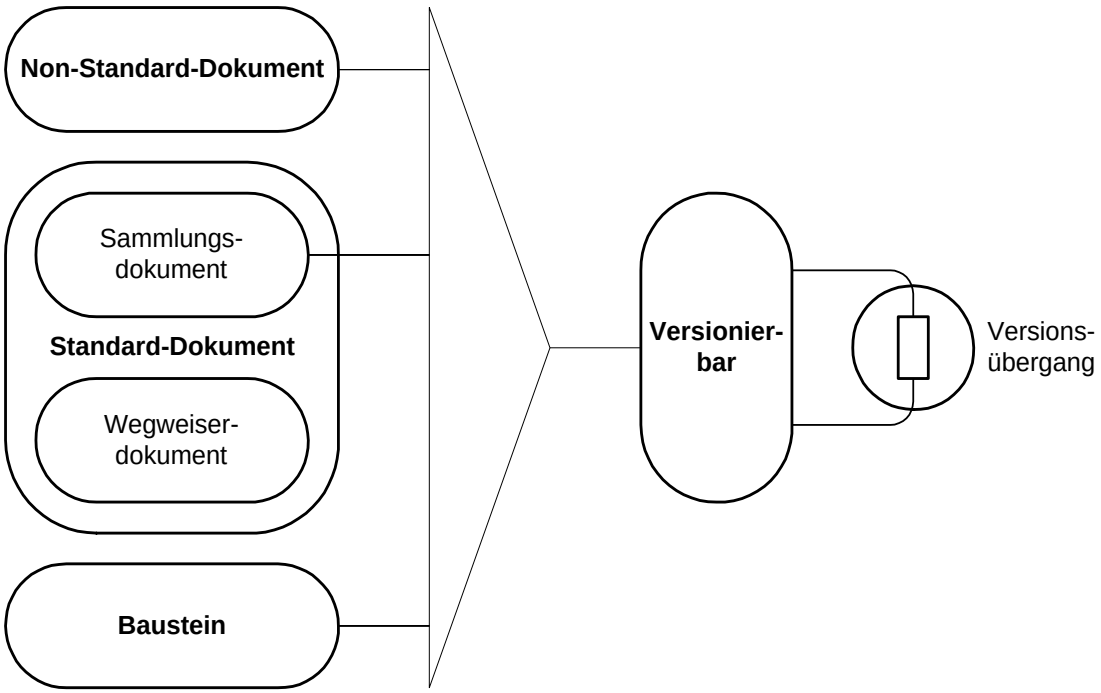


Bild 9: ER-Diagramm zur Versionsstruktur

Jede versionierbare Entität kann höchstens einen Nachfolger und einen Vorgänger besitzen, dies ist durch die Relation auf die Menge aller versionierbaren Elemente dargestellt. Die Objektifizierung der Relation, die im folgenden als ***Versionsübergang*** bezeichnet wird, ist durch die Kommentierung des Versionsübergangs zu begründen. Dies wird jedoch ausführlicher im Zusammenhang mit den Kommentaren (Abschnitt 4) erklärt.

3. Ablage, Transportwesen und Container

Nachdem der vorhergehende Abschnitt die Elemente, über die die Information in das Corporate Memory eingeht, behandelt hat, sollen nun die zur Aufbewahrung dieser Elemente nötigen Strukturen und Prozesse besprochen werden.

3.1 Ablage

Der Benutzer des Dokumentenverwaltungssystem erkennt, daß die Informationsträger in zwei Bereichen aufbewahrt werden. Der erste Bereich ist öffentlich und wird im folgenden *Bibliothek* genannt. Der zweite Bereich ist privat und wird als *Büro* eines Benutzers bezeichnet. Für jeden Benutzer gibt es mindestens ein Büro. Die Bibliothek und die Büros der Benutzer sind die für den Benutzer sichtbaren Aufbewahrungsräume der Informationsträger. Diese Aufbewahrungsräume und die in ihnen befindlichen Gegenstände (Bücher, Videokassetten usw.) bilden den Teil des Dokumentenverwaltungssystems, der als ***Ablage*** bezeichnet wird.

In einer Bibliothek ist es möglich, daß mehrere Exemplare eines Buches in verschiedenen Regalen stehen. Dies ist sinnvoll, um eine Abgeschlossenheit der Themengebiete zu gewährleisten, da manche Bücher mehreren Themengebieten zugeordnet werden können. Technisch bedeutet das, daß Gegenstände der Ablage Stellvertreter der abgelegten Informationsträger sind, über die auf die Informationsträger zugegriffen werden kann. Gegenstände der Ablage werden deshalb als ***Dokumentendummy*** bzw. ***Bausteindummy*** bezeichnet, da sie Stellvertreter von Dokumenten und Bausteinen sein können.

Bei den Aufbewahrungsräumen der Ablage handelt es sich eigentlich um Behälter. Ein Gebäude kann z. B. mehrere Säle beinhalten, ein Regal kann Behälter mehrerer Gegenstände sein. Für diese Behälter ist es genauso wie bei den Gegenständen sinnvoll, mehrere Exemplare ein und desselben Behälters zu haben. Beispielsweise sollte es möglich sein, daß ein Regal, das alle Bücher über den Bereich der Digitalen Systeme enthält, im Saal der Elektrotechnik und im Saal der Informatik zu finden ist. Für beide Bereiche ist ein solches Regal sinnvoll. Wünschenswert wäre nun, daß der Inhalt beider Regale identisch ist. Um dies zu erreichen, muß jeder Gegenstand, der in eines der Regale hineingestellt bzw. aus einem der Regale genommen wird, ebenfalls in das andere hineingestellt bzw. aus dem anderen genommen werden. Technisch ist dies zu lösen, indem jedes Regal nur ein Stellvertreter eines Behälterelements ist. Das bedeutet, daß genauso wie die Gegenstände auch die Aufbewahrungsräume nur Stellvertreter sind, über die auf die eigentlichen Behälterelemente zugegriffen wird. Die Behälter werden im folgenden als ***Container*** und die Aufbewahrungsräume der Ablage als ***Containerdummy*** bezeichnet.

Um die beschriebenen Konzepte zu veranschaulichen, stelle man sich ein Stockwerk mit zwei Sälen vor. In den Sälen befinden sich jeweils zwei Regale mit Büchern. Bild 10 zeigt eine Gesamtansicht dieser Ablage. Die abgerundeten Rechtecke sind die Aufbewahrungsräume, die anderen Rechtecke die Gegenstände. Die Aufbewahrungsräume sind durch verschiedene

Graustufen zu unterscheiden. Jede Graustufe entspricht einem Aufbewahrungstyp. Das Stockwerk ist hellgrau, jeder Saal mittelgrau und ein Regal dunkelgrau dargestellt. Elemente der gleichen Form haben den gleichen Inhalt.

Auffallend bei dem Beispiel ist, daß Regal II und III Exemplare ein und desselben Regals sind. Sie haben gleiche Form und damit gleichen Inhalt. Die zehn Gegenstände, die in den Regalen enthalten sind, sind Exemplare von sechs unterschiedlichen Dokumenten bzw. Bausteinen.

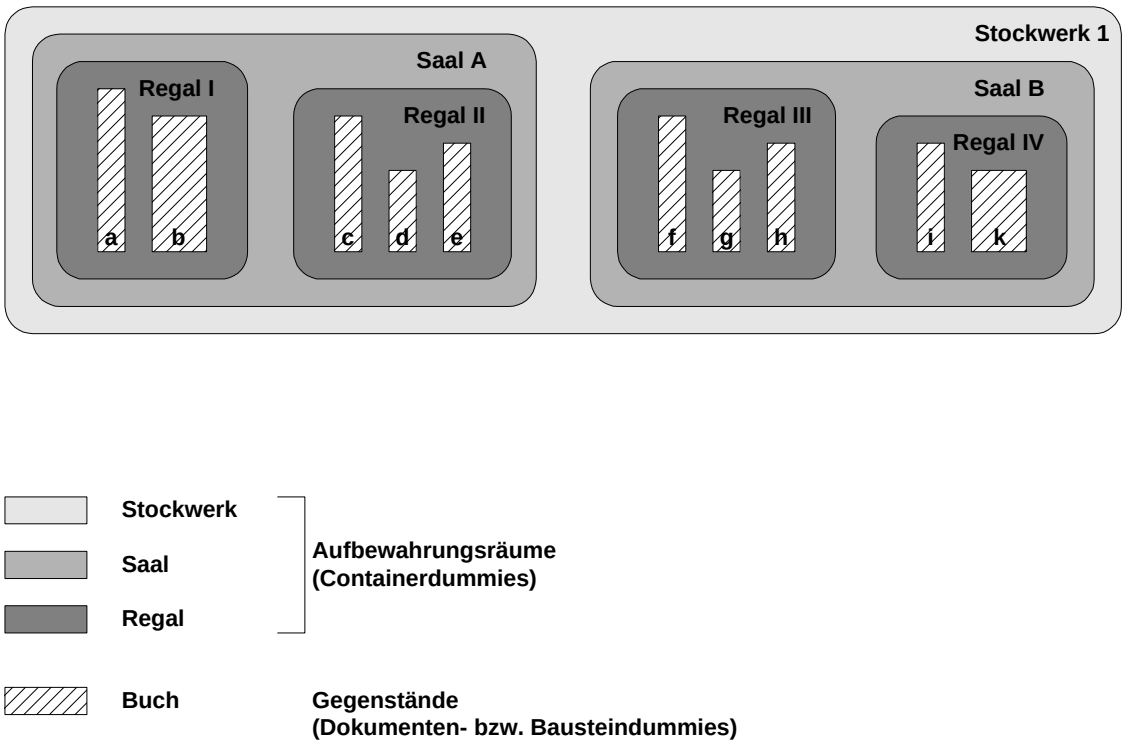


Bild 10: Ablagebeispiel

Die Ablage ist baumstrukturiert, wobei das Stockwerk die Wurzel und die Bücher die Blätter der Struktur sind. Dies ist im Strukturbaum in Bild 11 zu sehen. Dabei ist zu erkennen, daß jeder der Aufbewahrungstypen den gleichen Abstand von der Wurzel hat.

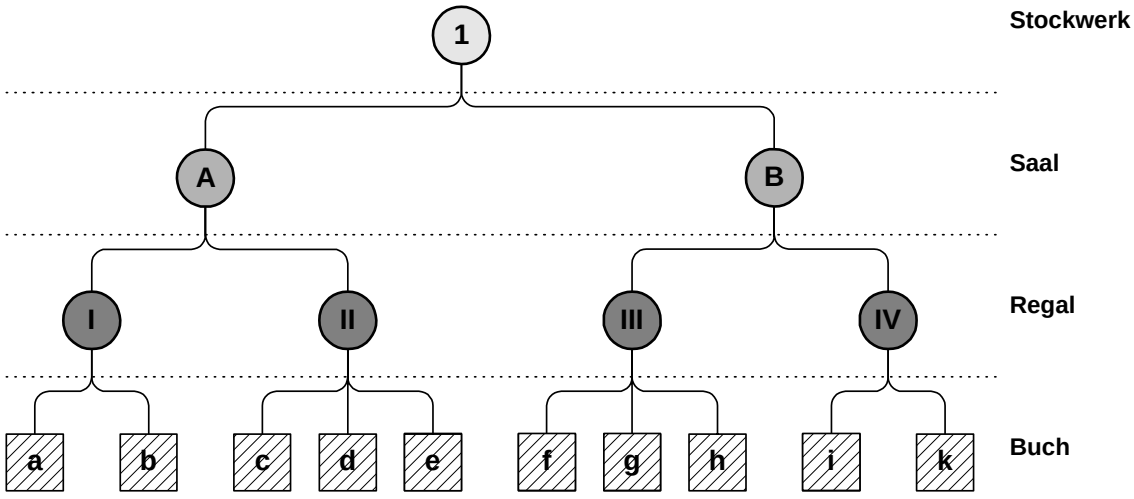


Bild 11: Baumstruktur der Beispielablage

Allerdings ist es nicht nur nötig, daß auf Container, Dokumente und Bausteine über Stellvertreter aus der Ablage zugegriffen werden kann. Die Container müssen ebenso Ablageelemente enthalten können. Dadurch entsteht die Struktur in Bild 12, in dem Container als Kreise, Dokumente und Bausteine als Rechtecke modelliert werden. Die Stellvertreter haben die gleiche Form, sind aber grau bzw. schraffiert dargestellt.

Da der Benutzer nur über Stellvertreter auf die Originale zugreifen kann, ist das erste Element der Stellvertreter, der auf den Stockwerkscontainer verweist. In diesem sind die Stellvertreter der Säle enthalten, die in dem Stockwerk liegen. In den Saalcontainern befinden sich Stellvetreter der Regale. Durch die Stellvertreter ist es möglich, daß man von Saal A und Saal B über unterschiedliche Stellvertreter zu dem gleichen Regalcontainer gelangt. Damit ist es möglich, daß der Benutzer, der die zweistufige Struktur von Stellvertreter und Original nicht wahrnimmt, den Eindruck gewinnt, daß sich Regal II und III an unterschiedlichen Orten befinden, und trotzdem immer den gleichen Inhalt haben. Durch die gleiche Methode ist es möglich, daß Buch V und VI an verschiedenen Orten in der Ablage stehen, aber nur auf ein Buch (Dokument) zugegriffen wird.

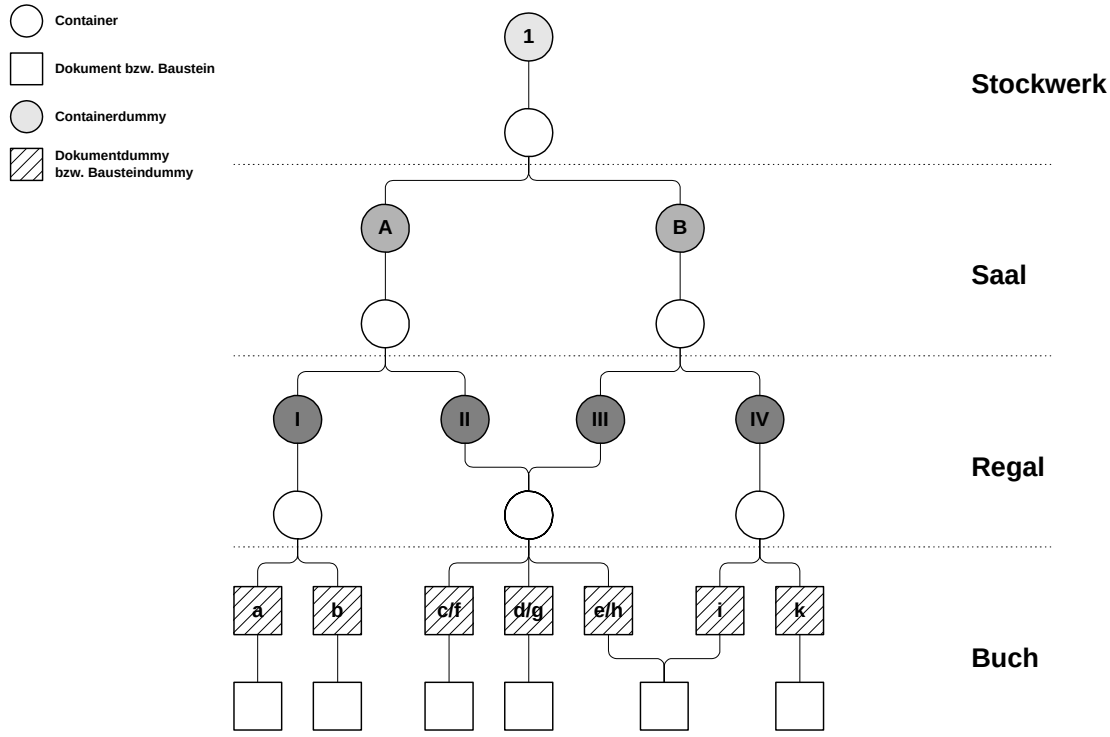


Bild 12: Archivierungsstruktur

Aus diesen Überlegungen heraus läßt sich das ER-Diagramm zur Ablage ableiten. Dies ist in Bild 13 zu sehen. Jeder Container hat mindestens einen Stellvertreter, der ihn in der für den Benutzer erfaßbaren Ablage vertritt, d.h. es gibt mindestens einen Containerdummy, der auf einen Container verweist. Alle Stellvertreter, außer dem Stellvertreter des Wurzelcontainers müssen in einem Container liegen. Um Zyklen in der Struktur zu vermeiden, darf ein Containerdummy nicht auf einen Container verweisen, dessen Wurzelabstand kleiner oder gleich dem Wurzelabstand des Containers ist, in dem er liegt. Im Beispiel heißt das, daß ein Contai-

nerdummy, der in einem Saalcontainer liegt, nicht auf einen anderen Saalcontainer bzw. auf den Stockwerkscontainer verweisen darf. An einem Containerdummy kann ein Wegweiserdokument hängen, das Zusatzinformationen von den Erstellern der Ablage enthält.

Dokumente und Bausteine sind über eine möglicherweise leere Menge von Dummies in der Ablage vertreten. Verweist kein Dummy auf ein Dokument bzw. Baustein, dann ist es nicht direkt über die Ablage erreichbar. Ein Dokument darf nur dann keinen Stellvertreter besitzen, wenn sichergestellt ist, daß es Komponente eines anderen Standard-Dokuments ist, auf das aus der Ablage zugegriffen werden kann. Beispielsweise ist ein Textdokument, daß alleine keinen Sinn ergibt, Teil eines größeren Dokuments. Es ist nicht sinnvoll, daß dieses Textdokument unter einem eigenen Namen in der Bibliothek vorhanden ist. Durch die Einbettung in ein großes Dokument, das in der Bibliothek steht, ist es aber jederzeit indirekt über dieses zu erreichen. Bei Bausteinen ist der Normalfall, daß sie als Typ eines Bausteinexemplars indirekt erreichbar sind. Nur wenn ein Baustein als Vorlage abgelegt wird, ist er direkt, also über einen Stellvertreter, zu erlangen.

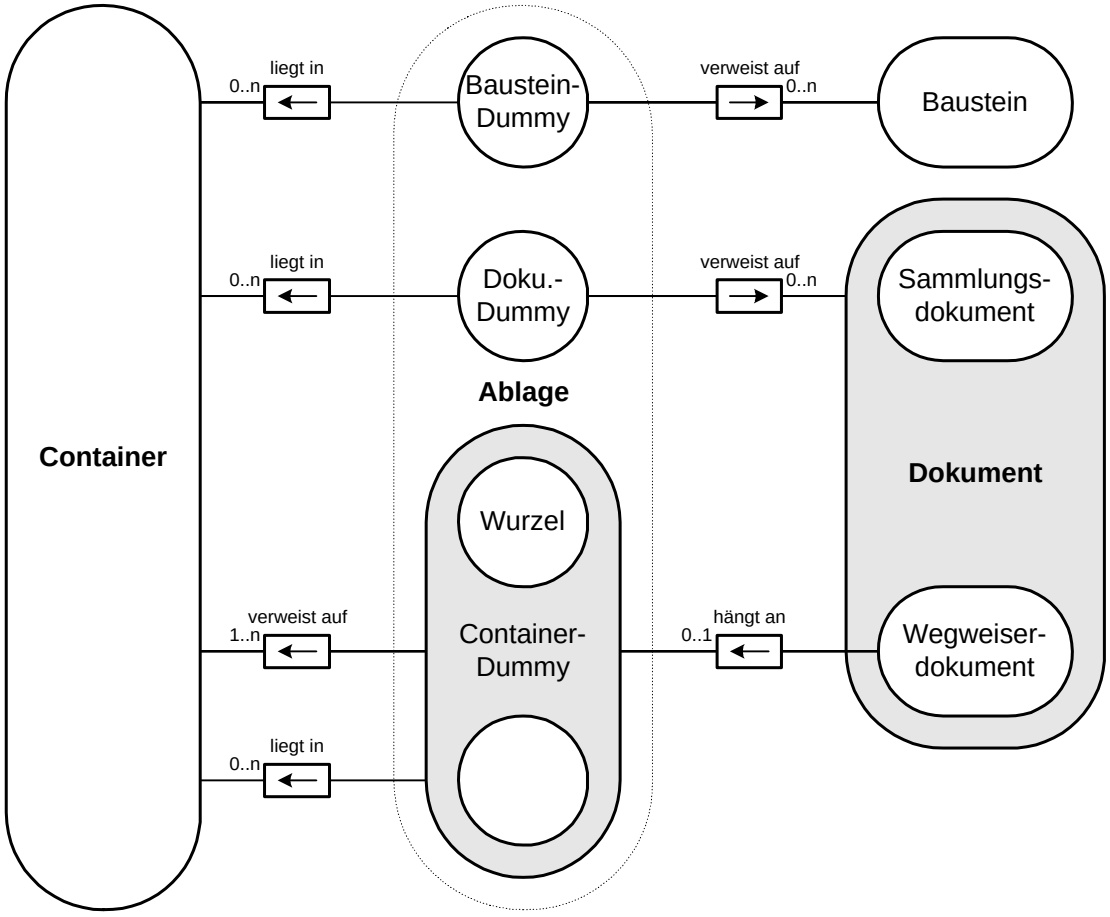


Bild 13: ER-Diagramm der Ablagestruktur

3.2 Transportwesen

Das Transportwesen der Dokumentenverwaltung ist so gewählt, daß es starken Einfluß auf die Struktur der Verwaltungsdaten hat. Aus diesem Grund behandelt der folgende Abschnitt diese näher.

Um das Transportwesen besser zu verstehen, wird es an einem Beispiel verdeutlicht. Man stelle sich vor, man befindet sich im privaten Bereich der Ablage, also im Büro. Ein Artikel, der gerade fertiggestellt wurde, soll veröffentlicht werden, d.h. der Artikel soll vom Schreibtisch des Büros in die Bibliothek transportiert werden. Solch ein Transport erfolgt in zwei Phasen. Zuerst wird der Artikel durch einen Transportakteur in einen sogenannten ***Eingangskorb*** (Inbox) innerhalb der Bibliothek befördert. In der zweiten Phase wird der Artikel von einem Bibliothekar aus dem Eingangskorb genommen und in die Bibliothek eingeordnet.

Solche Eingangskörbe können sich in jedem Saal der Bibliothek und in jedem Büro befinden, da diese für Menschen überschaubare Einheiten, darstellen.

Aufgabe des Transportwesens ist es, die erste Phase des Transports durchzuführen. Die Beschreibung des Transportwesens läßt sich dabei in zwei Themengebiete gliedern.

- Transportelement
- Transportvorgang

3.2.1 Transportelement

Das vom Transportakteur transportierte Element ist ein Paket. Das Paket besteht aus einem Transportbehälter und einem Begleitpapier. Der Transportbehälter enthält Elemente der Ablage und wird als ***Transportcontainer*** bezeichnet. Das Begleitpapier beinhaltet die für den Transporteur und den Empfänger erforderlichen Transportinformationen. Das Begleitpapier wird auch als ***Transportformular*** bezeichnet, da es nichts anderes als ein ausgefülltes Formular zur Durchführung eines Transports ist.

In Bild 15 auf Seite 26 ist die Struktur eines solchen Pakets zu erkennen. Es handelt sich dabei um eine strukturierte Entität, die links unten im Bild zu sehen ist. Der Transportcontainer eines Pakets kann wie ein normaler Ablagecontainer alle Elemente bis auf die Wurzel der Ablage enthalten. Dadurch ist es möglich, außer Gegenständen auch Aufbewahrungselemente zu transportieren. An einem Transportcontainer hängt immer ein Transportformular. Dieses muß jedoch nicht immer an einem Transportcontainer hängen. Es kann auch in einem Formularcontainer aufbewahrt werden. Wichtig ist, daß ein Transportformular entweder an einem Transportcontainer hängt oder in einem Formularcontainer liegt. Dies ist durch die Zusatzbedingung im Bild links unten ausgedrückt. Um den Ort, an dem ein Paket abgelegt werden soll, anzugeben, verweist das Transportformular auf ein Aufbewahrungselement.

3.2.2 Transportvorgang

Hat derjenige, der ein Ablageelement verschicken will, dieses mittels eines Pakets transportfähig gemacht, gibt er dem Transportakteur den Auftrag, es zu transportieren. Der Transportakteur versucht dann, das Paket in ein als Eingangskorb gekennzeichnetes Aufbewahrungselement zu legen. Der Eingangskorb, in den das Paket plaziert, wird durch den Auftraggeber und die ***Transportstruktur*** der Ablage bestimmt. Der Auftraggeber verweist durch das Formular auf ein Aufbewahrungselement. Er schlägt damit den Bereich der Ablage vor, in den das Paket transportiert werden soll. Vorteil dieser Vorgehensweise ist, daß der Auftraggeber nicht wissen muß, wer diesen Bereich der Ablage pflegt. Das Paket wird automatisch in den richtigen Eingangkorb gelegt. Dies geschieht aufgrund der Transportstruktur.

In Bild 14 ist die Transportstruktur der in Bild 10 auf Seite 20 dargestellten Ablage gezeigt.

In jedem Saal befindet sich eine Inbox, auf die von den zugehörigen Sälen gedeutet wird. Die in den Sälen liegenden Regale verweisen auf die sie beinhaltenden Säle. Regal I und III zeigen zusätzlich auf das jeweils andere Regal hin. Diese in der Darstellung gestrichelten Pfeile zwischen Regal I und Regal II machen darauf aufmerksam, daß der Verweis erst dann verfolgt werden soll, wenn die durch den Hauptpfeil erreichbare Inbox voll ist. Dies dient der Verteilung der Pakete bei Überlastung eines Bibliothekars.

Ist der Bibliothekar von Saal A in Urlaub, kann von seiner Inbox auf die des Kollegen, der Saal B pflegt, verwiesen werden. Pakete, die an das Stockwerk adressiert sind, werden an die Inbox III weitergeleitet. Diese könnte beispelsweise durch beide Bibliothekare gepflegt werden.

Die nötigen Strukturen, um solche Transporte zu ermöglichen, werden durch die “verweist auf”-Relation zwischen Containerdummies (in Bild 15 auf Seite 26 rechts unten) gewährleistet. Dadurch, daß jeder Containerdummy auf eine beliebige Menge von Containerdummies verweisen kann und auf einen Containerdummy beliebig oft verwiesen werden darf, ist jede Transportstruktur möglich.

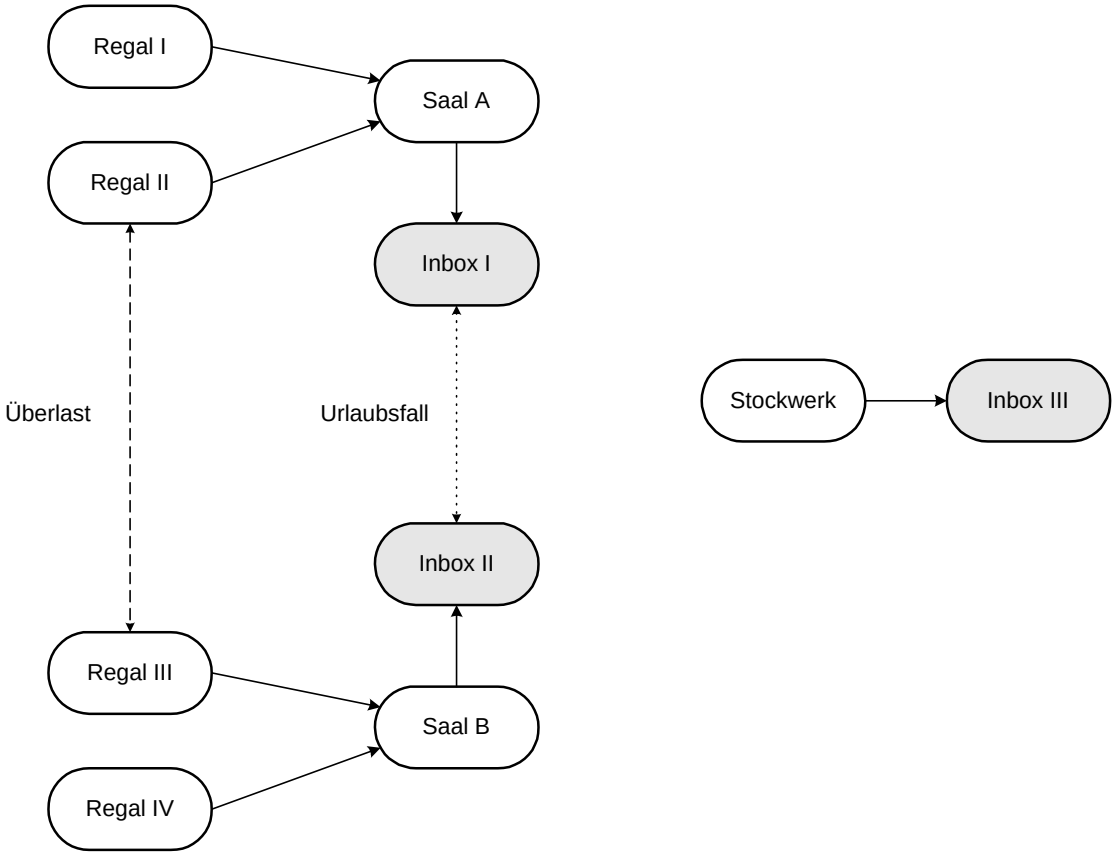
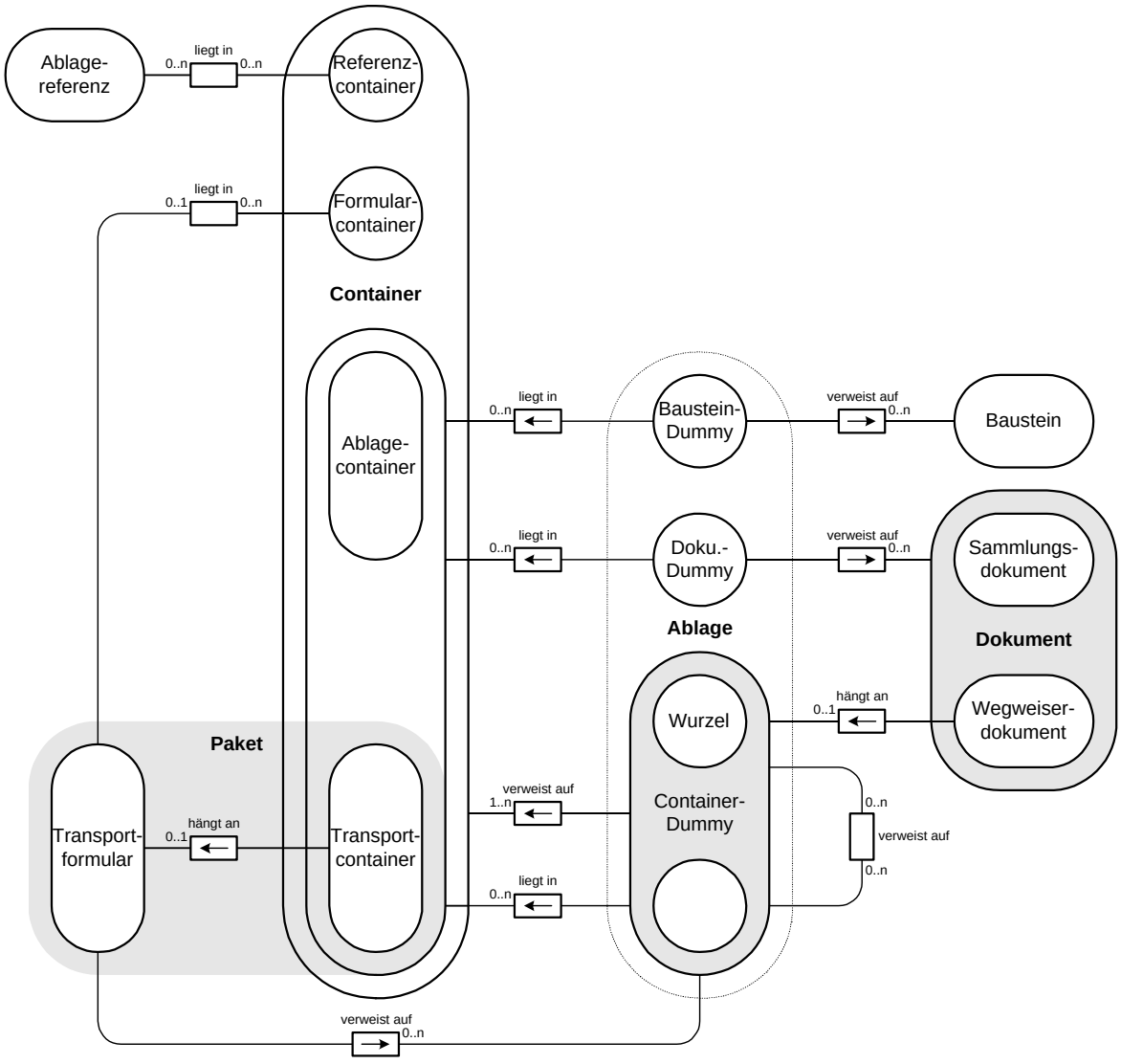


Bild 14: Transportstruktur

3.3 Container

Da ein Container Behältercharakter hat, kann dieser nicht nur Ablageelemente zur Strukturierung der Ablage enthalten, sondern auch andere Elemente. Um eine Trennung des Inhalts der Container zu gewährleisten, gibt es vier Containertypen, die in Bild 15 links in der Klasse der Containerentitäten enthalten sind.



$\forall$Transportformular: [(Transportformular, Formularcontainer) $\in$ liegt in] $\vee$
[(Transportformular, Transportcontainer) $\in$ hängt an]

Bild 15: ER-Diagramm der Ablage- und Containerstruktur

Die Ablagecontainer werden wie folgt unterschieden:

Ablagecontainer
Der Ablagecontainer enthält nur Ablageelemente. Dies ist der Containertyp, der der Strukturierung der Ablage dient.

Transportcontainer
Der Transportcontainer enthält, ebenso wie der Ablagecontainer, Ablageelemente. Er bildet, wie im vorausgegangenen Abschnitt erläutert, das zu Transportzwecken benötigte Paket.

Referenzcontainer
Der Referenzcontainer kann Ablagereferenzen enthalten. Die Ablagereferenzen können aufbewahrt werden, um sich Orte in der Ablage zu merken. Dadurch kann ein Benutzer Referenzen auf häufig benutzte Regale in der Bibliothek ablegen, um diese ohne Umwege zu erreichen. Eine ausführlichere Beschreibung der Referenzen erfolgt in Abschnitt 5.

Formularcontainer
Der Formularcontainer enthält Transportformulare. Das Aufbewahren dieser Formulare kann zum einen zur Kontrolle und Analyse von transportierten Paketen dienen. Zum anderen besteht die Möglichkeit, den Formularcontainer als Ablage für Transportformulare zu nutzen, die als Vorlage zur Erstellung standardisierter Transportpakete benötigt werden. Beispielsweise ist es zweckmäßig, ein Transportformular eines Pakets aufzuheben, das dem Transport eines Berichts diente. Bei erneutem Erstellen eines Pakets für den Transport eines Berichts kann das Transportformular wiederverwendet werden.

3.4 Attribute

3.4.1 Ablage

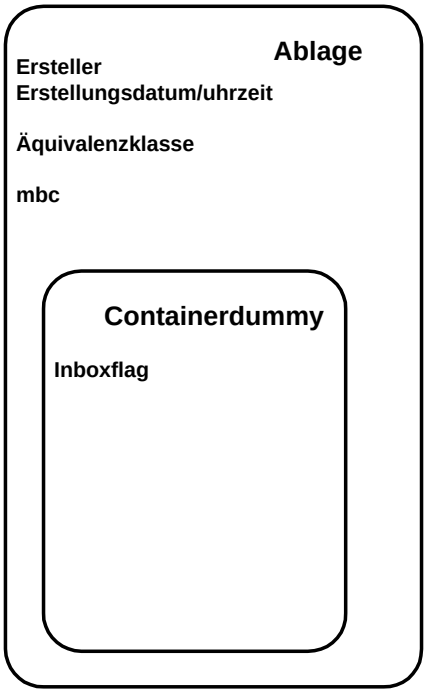


Bild 16: Ablageattribute

Erstellung

- **Ersteller:**
  Ersteller des Ablageelements.
- **Erstellungsdatum/uhrzeit:**
  Datum und Uhrzeit der Erstellung der Ablage.

Änderung

- **Änderer:**
  Letzte Person, die das Ablageelement verändert hat.
- **Änderungsdatum/uhrzeit:**
  Datum und Uhrzeit der letzten Änderung.

Berechtigung

- **Äquivalenzklasse:**
  Klasse, über die die Zugriffsrechte auf das Ablageelement abgeleitet werden.

Performance

- **mbc:**
  Durchschnittliche zu erwartende Zeitspanne zwischen Änderungen eines Ablageelements.

Transport

- **Inboxflag:**
  Ist das Inboxflag gesetzt, dann kann der Container als Eingangskorb benutzt werden.

3.4.2 Transportwesen

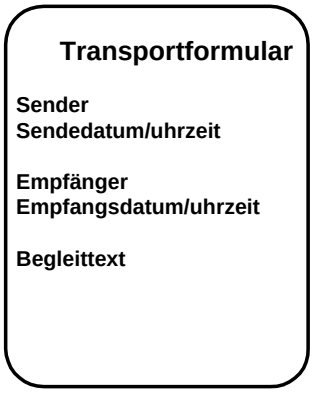


Bild 17: Transportformularattribute

Sendung

- **Sender:**
  Absender eines Pakets.
- **Sendedatum/uhrzeit:**
  Datum und Uhrzeit der Sendung eines Pakets.

Empfang

- **Empfänger:**
  Empfänger eines Pakets.
- **Empfangsdatum/uhrzeit**
  Datum und Uhrzeit des Empfangs eines Pakets.

Beschreibung

- **Begleittext:**
  Zusatzinformation für den Empfänger des Pakets.

3.4.3 Container

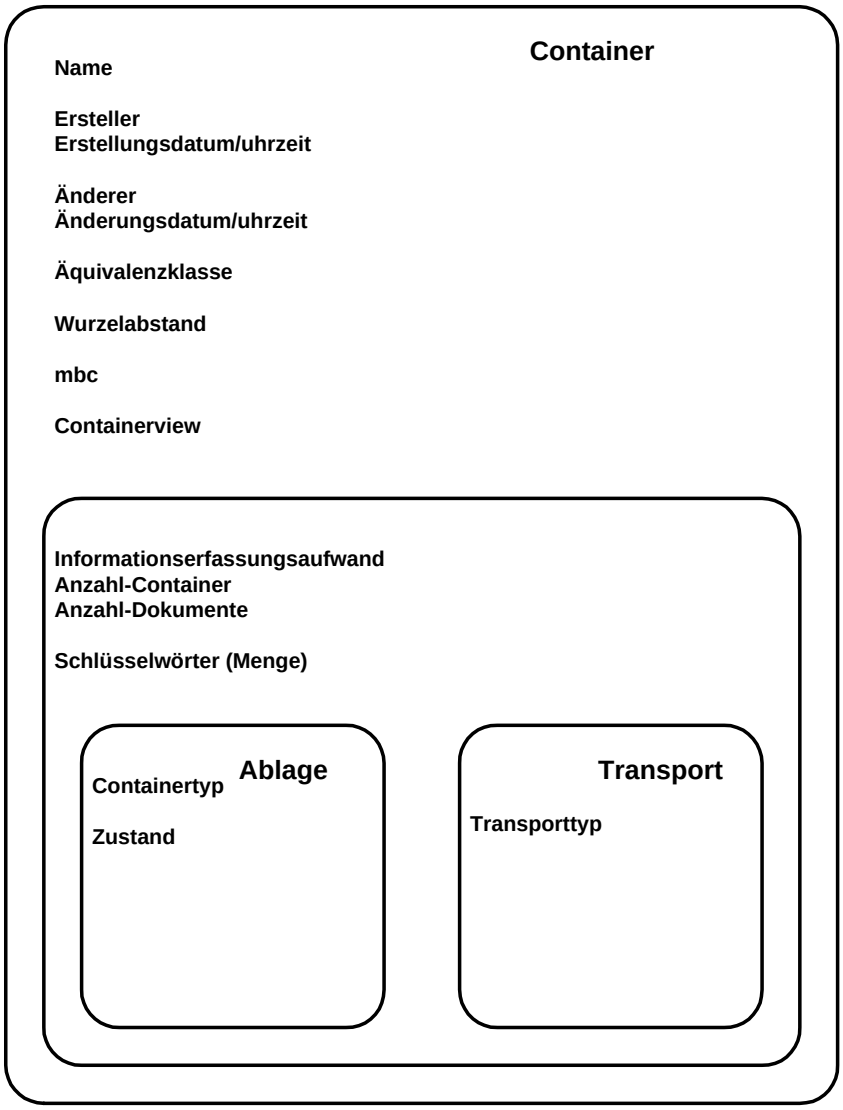


Bild 18: Containerattribute

Identifikation

- **Name**
  Name des Containers.

Erstellung

- **Ersteller:**
  Ersteller des Containers.
- **Erstellungsdatum/uhrzeit:**
  Datum und Uhrzeit der Erstellung des Containers.

Änderung

- **Änderer:**
  Letzte Person, die den Container verändert hat.
- **Änderungsdatum/uhrzeit:**
  Datum und Uhrzeit der letzten Änderung.

Berechtigung

- **Äquivalenzklasse:**
  Klasse, über die die Zugriffsrechte auf den Container abgeleitet werden.

Ordnungshierachie

- **Wurzelabstand:**
  Abstand des Containers vom Wurzelcontainer.

Performance

- **mbc:**
  Durchschnittliche zu erwartende Zeitspanne zwischen Änderungen des Containers.

Darstellung

- **Containerview:**
  Darstellungsart eines Containers in der Ablage.

Inhalt

- **Informationserfassungsaufwand:**
  Gesamter Informationserfassungsaufwand der Dokumente bzw. Bausteine, die durch den Container erreichbar sind.
- **Anzahl-Container:**
  Gesamte Anzahl der Container, die durch den Container erreichbar sind.
- **Anzahl-Dokumente:**
  Gesamte Anzahl der Dokumente bzw. Bausteine, die durch den Container ereichbar sind.

Beschreibung

- **Schlüsselwörter:**
  Zur Unterstützung der Suche nach Dokumenten kann ein Container eine möglicherweise leere Menge an Schlüsselwörter haben.

Typ

- **Containertyp:**

  Typ des Containers. Vorgesehene Typen eines Ablagecontainers sind beispielsweise Land, Campus, Haus usw.[1] .

- **Transporttyp:**

  Typ des Transportcontainers. Es gibt drei Transportcontainertypen: Klein, Mittel, Groß. Durch die Transporttypen kann der maximale Inhalt der Transporcontainer bestimmt werden.

Klassifikation

- **Zustand:**

  Ein Ablagecontainer kann sich in drei Zuständen befinden. Der erste ist der der *normalen Benutzung*. Der Container wird von den Bibliothekaren bezüglich seines Inhalts ständig aktualisiert. Ist der Container im Zustand *eingefroren*, dann wird der Container nur noch als Übergangscontainer benutzt. Ein Übergangscontainer ist ein zur Zerstörung vorgesehener, Container der für eine Übergangszeit im Dokumentenverwaltungssystem verbleibt. Die Übergangszeit ist für einen sanften Übergang bei einer Umstrukturierung gedacht. Befindet sich der Container im dritten Zustand, dann wird ein Container gewartet. Der Zustand *Wartung* ist für den Fall vorgesehen, daß ein Bibliothekar einen Containerinhalt umorganisiert und es nicht sinnvoll ist, den Benutzern zu erlauben, auf den Inhalt des Containers zuzugreifen.

1. Eine genauere Beschreibung dieser Typen erfolgt in der Arbeit von Christian Brand [Brand 97].

4. Kommentare

Um Anmerkungen zu Elementen des Dokumentenverwaltungssystems machen zu können, ist es möglich, Kommentare zu erstellen und den Elementen zuzuordnen. Kommentare sind meist kurze Texte, deshalb sind sie statisch-flächige und unstrukturierte Informationsträger. Der Form nach entsprechen sie den Bausteinbasen und können wie diese verwaltete Referenzen beinhalten. Die Möglichkeit, daß Kommentare Referenzen und alles Druckbare enthalten können, ist ein Zugeständnis an die am Anfang des Kapitels gemachte Aussage, daß Strukturen länger existieren als Algorithmen. Bei dem geplanten Dokumentenverwaltungssystem sind nur Texte ohne Referenzen zugelassen. Um jedoch nicht im voraus die Erstellung anderer Kommentare als die rein textuellen zu verhindern, sind Referenzen und alles Druckbare bei Kommentaren zugelassen. Man unterscheidet drei Arten von Kommentaren.

4.1 Versionskommentar

Mit einem Versionskommentar wird ein Versionsübergang kommentiert. Der Autor eines Dokuments bzw. Bausteins kann damit Information zum Versionsübergang einbringen. Typisch hierfür ist der Grund der Versionierung oder Hinweise auf die Änderungen. Diese Information gehört nicht zum Dokument. Deshalb hängt der Versionsübergangskommentar am Versionsübergang und nicht am Dokument. An dem Versionsübergang darf dabei höchstens ein Kommentar hängen, wie in Bild 19 durch die „hängt an“-Relation zwischen Versionskommentar und Versionsübergang beschrieben ist.

4.2 Ablagekommentar

Mit einem Ablagekommentar wird ein Aufbewahrungselement kommentiert. Er dient in der Hauptsache den Benutzern der Bibliothek. Mit einem Ablagekommentar kann ein Benutzer dem Biliothekar Hinweise zur Strukturierung der Bibliothek geben. Beispielsweise kann ein Benutzer einen Kommentar an ein Regal hängen und dadurch den Bibliothekar, der das Regal pflegt, auf Strukturmängel aufmerksam machen. Ein Ablagekommentar hängt deshalb an einem Containerdummy, wie in Bild 19 durch die Relation zwischen Ablagekommentar und Containerdummy dargestellt ist.

4.3 Leserkommentar

Mit einem Leserkommentar wird ein Dokument kommentiert. Dabei gibt es drei Arten von Leserkommentaren, den Hinweis-, den Warnungs- und den Fehlerkommentar.

- *Hinweiskommentar*

  Bei einem Hinweiskommentar soll meist auf Mängel (aus Sicht des Lesers) innerhalb eines Dokuments hingewiesen werden.

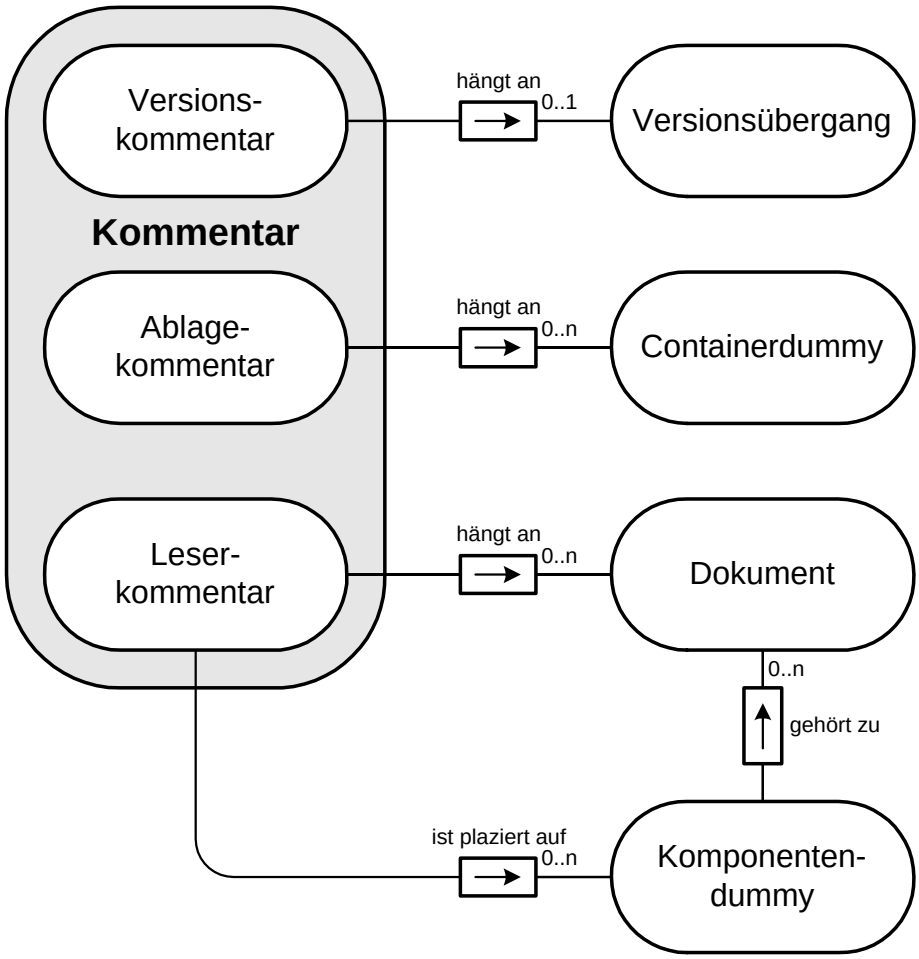


Bild 19: ER-Diagramm zur Kommentarstruktur

- *Warnungskommentar*

  Bei einem Warnungkommentar will der Leser den Autor des Dokuments auf mögliche Fehler aufmerksam machen. Diese Fehler sind unkritisch bzw. nicht eindeutig, deshalb ist der Kommentar nur eine Warnung. Ein typischer Fall für einen Warnungskommentar wäre bei einem in Form eines Dokuments abgelegten Sourcecode angebracht, der fehlerhaft, aber für den momentanen Gebrauch akzeptabel ist.

- *Fehlerkommentar*

  Bei einem Fehlerkommentar will der Leser den Autor auf Fehler im Dokument hinweisen. Beispiel für den Gebrauch eines Fehlerkommentars wäre ein in Form eines Dokuments abgelegter Sourcecode, der einen schwerwiegenden Fehler enthält.

Die Einordnung, um welche Art von Leserkommentar es sich handelt, liegt im Ermessensspielraum des Kommentarerzeugers.

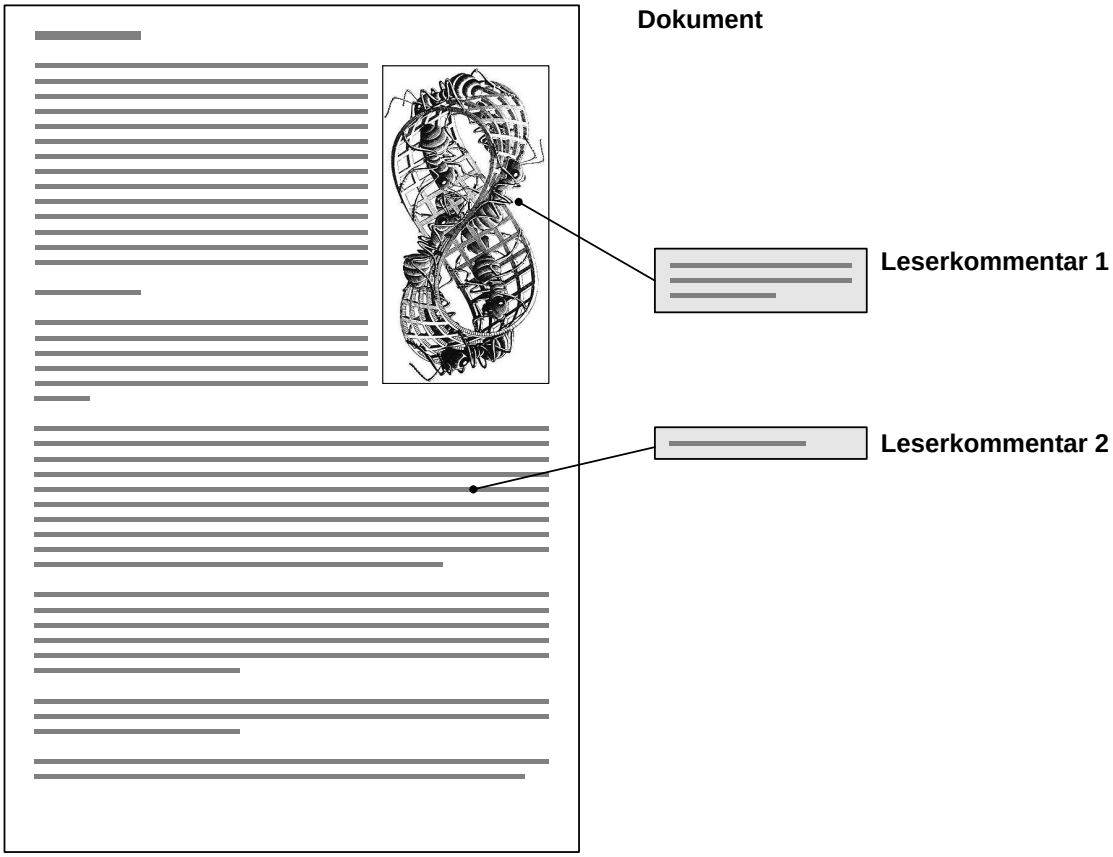


Bild 20: Leserkommentar auf einem Standard-Dokument

Ein Leserkommentar hängt immer an einem Dokument. Er gehört damit eindeutig zu diesem Dokument und zu keinem anderen. Wenn das Dokument Komponente eines anderen Dokument ist, erscheint der Kommentar nicht im zusammengesetzten Dokument. Ebenso taucht ein Kommentar, der in einem strukturierten Dokument auf eine Komponente plaziert wurde, nicht in dem als Komponente dienenden Dokument auf. Der Kommentar erscheint nur im strukturierten Dokument.

Das dargestellte Beispiel in Bild 20 und in Bild 21 soll den Sinn der Aussage verdeutlichen. An dem Dokument in Bild 20 hängen zwei Leserkommentare. Leserkommentar 1 ist auf einer Komponente des Dokuments plaziert und Leserkommentar 2 auf dem Wurzelbaustein. Obwohl Leserkommentar 1 auf der Komponente plaziert ist, kommentiert er nicht das Bild als eigenständiges Dokument, sondern im Kontext des Gesamtdokuments. Beispielsweise könnte der Kommentar wie folgt lauten: “Das Bild ist zwar schön, aber es hat nichts mit dem Text zu tun.”. Würde dieser Leserkommentar an dem Dokument in Bild 21 hängen, könnte der Autor dieses Dokuments nichts mit dem Kommentar anfangen. Genausowenig hätte der Autor des strukturierten Dokuments von dem Leserkommentar, der an dem Dokument in Bild 21 hängt, da dieser nur bildspezifisch ist. Der Kommentar könnte beispielsweise die unnatürliche Darstellung der Insekten bemängeln. Wenn aber das Dokument aus Bild 20 einen Text über die Technik des Tuschezeichnens enthält, dann ist dieser Kommentar dort sinnlos.

.

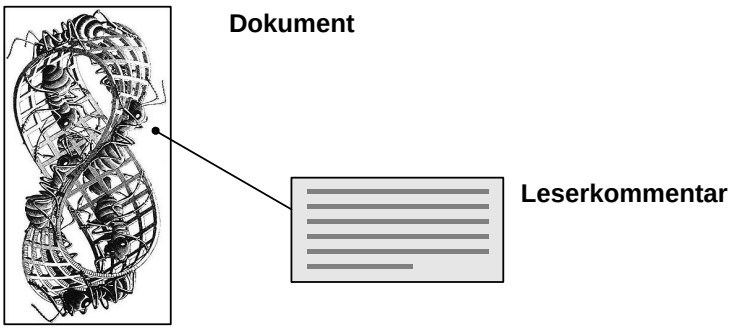


Bild 21: Leserkommentar auf einem bausteintauglichen Dokument

Die Struktur, die es ermöglicht, daß ein Leserkommentar auf einer Komponente plaziert ist, die zu einem strukturierten Dokument gehört, ist im ER-Diagramm in Bild 19 gezeigt.

Ein Leserkommentar hängt an einem Dokument und kann auf einem Komponentendummy plaziert sein, der zu einem Bausteinexemplar gehört, das Baustein des Dokuments ist, an dem der Kommentar hängt. Es ist damit möglich, daß auf einem Komponentendummy Leserkommentare plaziert sind, die an verschiedenen Dokumenten hängen.

4.4 Attribute

4.4.1 Kommentar

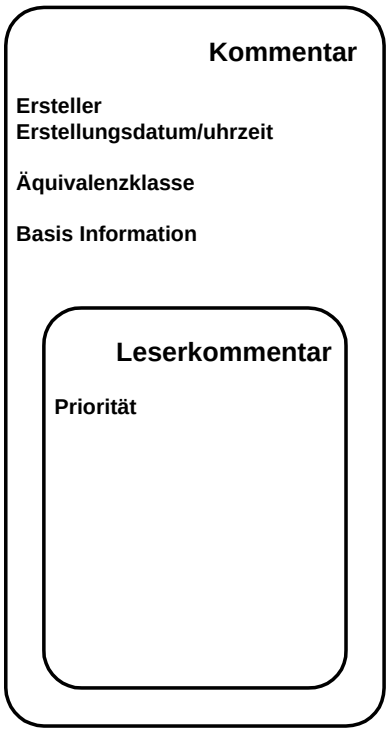


Bild 22: Attribute der Kommentare

Erstellung

- **Ersteller:**
  Ersteller des Kommentars.
- **Erstellungsdatum/uhrzeit:**
  Datum und Uhrzeit der Erstellung des Kommentars.

Berechtigung

- **Äquivalenzklasse:**
  Klasse, über die die Zugriffsrechte auf den Kommentar abgeleitet werden.

Verwaltete Information

- **Basisinformation:**
  Freilängenattribut, das die durch den Kommentar verwaltete Information beinhaltet.

Typ

- **Priorität:**
  Die Priorität gibt an, wie wichtig der Leserkommentar für den Autor des kommentierten Dokuments ist. Es gibt drei Prioritätsstufen: Hinweis, Warnung und Fehler. Die Bedeutung der einzelnen Prioritätsstufen wurde bereits in Abschnitt 4.3 über Leserkommentare beschrieben.

5. Referenz

Mit einer Referenz verweist ein Autor eines Dokuments auf etwas, das in Beziehung mit dem Inhalt des Dokuments steht. Referenzen können verwaltet oder nicht verwaltet sein. Bei verwalteten Referenzen unterstützt ein technisches System den Leser bei der Verfolgung der Referenz. Bei nicht verwalteten Referenzen stützt sich der Leser auf die Information, die in der Referenz zum Erreichen des Elements gegeben wurde.

Um eine Referenz zu verwalten, muß das Dokumentenverwaltungssystem das Format des Dokuments kennen. Deshalb gibt es verwaltete Referenzen nur bei Informationsträgern mit systemeigenem Format. Dies sind Kommentare, Standard-Dokumente und Bausteine. Da die eigentliche Information bei standard Dokumenten und Bausteinen in der Bausteinbasis liegt, befinden sich diese Referenzen in den Bausteinbasen.

Es gibt drei Arten von Referenzen, die im folgenden beschrieben werden.

5.1 Interne Referenz

Eine Interne Referenz ist eine Referenz, die auf eine markierte Stelle innerhalb der Bausteinbasis, in der sie liegt, verweist. Die markierten Stellen innerhalb einer Bausteinbasis werden als ***Anker*** bezeichnet. Diese können nur von deren Autoren erstellt werden, da die Markierung von Stellen ein Editionsvorgang ist.

Eine Bausteinbasis ist die Einheit, die ein Autor editiert. Um diese für den Leser überschaubar zu machen, kann der Autor eine interne Referenz verwenden.

Ein Kommentar ist ebenfalls eine von einem Autor editierbare Einheit. Er soll jedoch kurz und überschaubar gehalten werden, deshalb darf in Kommentaren keine interne Referenz liegen. Wenn ein Autor interne Referenzen verwenden müßte, wäre der Sinn des Kommentars verfehlt.

5.2 Dokumentreferenz

Eine Dokumentreferenz verweist auf ein abgelegtes Dokument. Ist das Dokument, auf das verwiesen werden soll, ein Standard-Dokument, kann die referenzierte Stelle innerhalb des Dokuments näher spezifiziert werden. Die Referenz kann auf einen Anker zeigen, der in dem Wurzelbaustein des referenzierten Dokuments liegt.

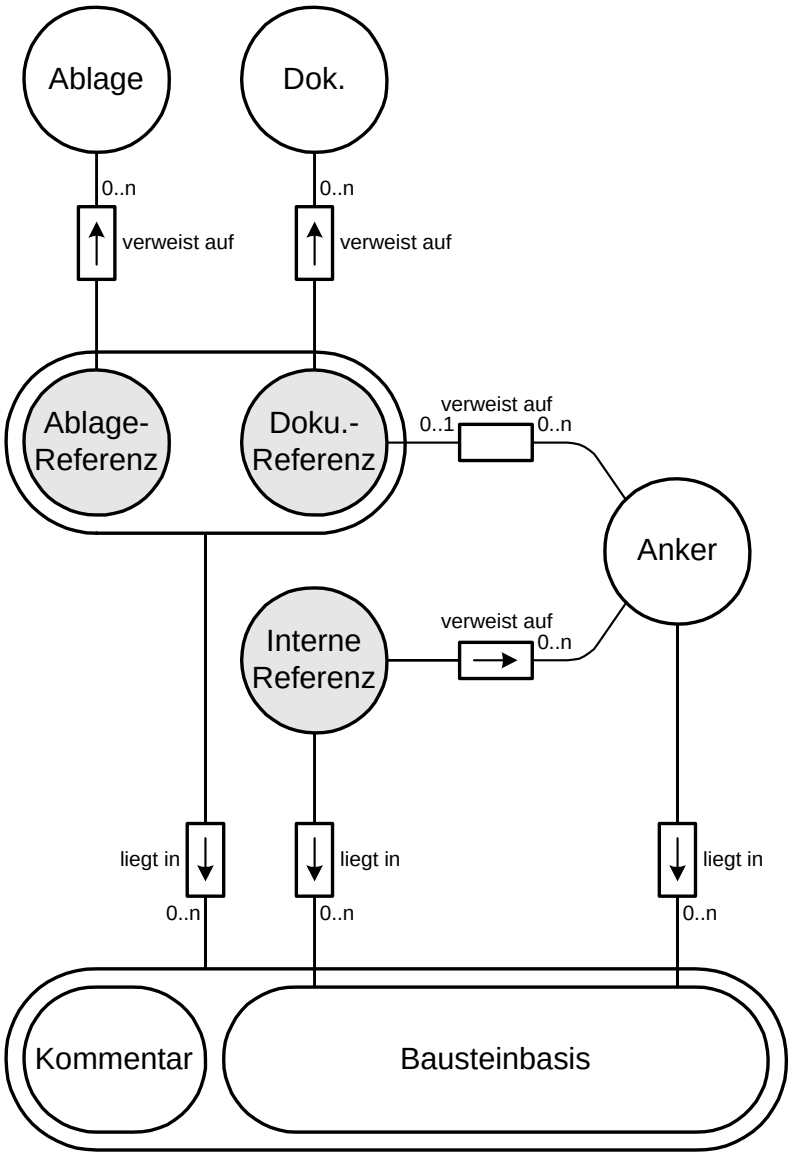
5.3 Ablagereferenz

Eine Ablagereferenz verweist auf ein Ablagedokument. Es ist dadurch möglich, direkt auf Gegenstände und Orte der Ablage zu verweisen. Dies kann für Wegweiserdokumente und Ablagekommentare nützlich sein, da der Inhalt dieser Dokumente sich immer auf die Ablage

bezieht. Für Versionskommentare und Sammlungsdokumente ist dies nicht sinnvoll, da deren Inhalt unabhängig von der Ablage sein soll.

5.4 Referenzstruktur

Aus den Abschnitten 4.1-4.3 folgt das in Bild 23 dargestellte ER-Diagramm. Die Referenzen sind die grauen Entitäten in der Bildmitte. Ablage- und Dokumentreferenzen liegen in Kommentaren oder in Bausteinbasen. Interne Referenzen liegen, wie bereits erläutert, nur in einer Bausteinbasis. Das gleiche gilt für die als Anker bezeichnete Markierung einer Stelle der Bausteinbasis. Auf einen Anker können interne Referenzen und Dokumentreferenzen verweisen.



$\forall$Anker, Bausteinbasis,Interne Referenz: [(Interne Referenz, Bausteinbasis) $\in$ liegt in] $*$ [(Interne Referenz, Anker) $\in$ verweist auf] $\rightarrow$ [(Anker, Bausteinbasis) $\in$ liegt in]

Bild 23: ER-Diagramm der Referenzstruktur

Interne Referenzen müssen immer auf einen Anker verweisen. Dokumentreferenzen können auf einen Anker verweisen, wenn das referenzierte Dokument ein Standard-Dokument ist. Interne Referenzen müssen, wie dies im Prädikat rechts unten im Bild gezeigt ist, immer in dem selben Baustein liegen, wie der Anker, auf den sie verweisen.

5.5 Attribute

5.5.1 Referenz

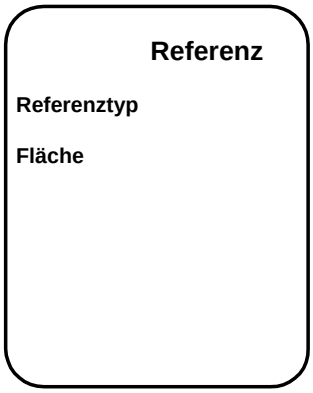


Bild 24: Attribute der Referenzen

Typ

- **Referenztyp:**
  Typ der Referenz.

Bereich

- **Fläche:**
  Bereich in einem Dokument, für den die Referenz gültig ist.

5.5.2 Anker

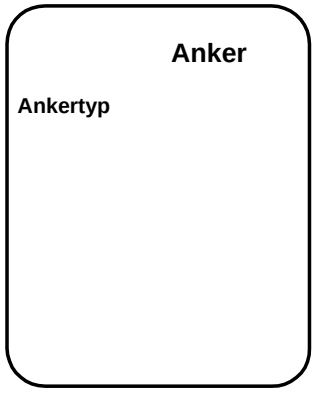


Bild 25: Attribute der Anker

Typ

- **Ankertyp:**
  Typ eines Ankers.

III Architekturüberlegungen

Die Hauptaufgabe dieser Arbeit bestand aus der Entwicklung der Struktur der Verwaltungdaten. Die Architekturüberlegungen wurden erst zum Ende der Arbeit angestellt. Aus diesem Grunde stellt dieses Kapitel nur den aktuellen Stand der Entwicklung dar und kann deshalb noch nicht als abgeschlossen betrachtet werden.

1. Datenbanksystem

Der in Bild 1 auf Seite 2 vorgestellte Aufbau des Dokumentenverwaltungssystems zeigt einen Dokumentenverwalter, der auf persistente Daten zugreift. Die Verwaltung solcher Daten ist zu komplex, um von dem Dokumentenverwalter übernommen zu werden.

Der Dokumentenverwalter operiert mit Daten in einem lokalen Speicher und kommuniziert mit einem Datenbankverwalter. Dieser übernimmt die Verwaltung der persistenten Daten. Die Daten werden in einer Datenbank gespeichert, auf die der Datenbankverwalter zugreift. Datenbankverwalter und Datenbank werden als ***Datenbanksystem*** bezeichnet. Bild 26 zeigt den allgemeinen Aufbau einer Anwendung, die auf ein Datenbanksystem zugreift.

Die Anwendung nutzt die vom Datenbankverwalter angebotenen Dienste über eine festgelegte Schnittstelle. Der Speicher der Anwendung enthält transiente (flüchtige) und persistente (dauerhafte) Daten.

Wie der Transport der persistenten Daten zwischen Datenbank und dem Speicher der Anwendung erfolgt, hängt vom Datenbankverwalter ab. Dieser kann einerseits direkten Zugriff auf die persistenten Daten des Anwendungsspeichers haben, andererseits kann die Anwendung diese Daten über den Datenbank-Bus an den Datenbankverwalter liefern.

Ein Datenbanksystem hat den Vorteil gegenüber herkömmlichen Dateisystemen, daß es die Beherrschung großer Datenbestände, Fehlerbehandlung und Verteilung der Daten gewährleistet. Ebenso bietet es Lösungen zur Wahrung der Datenintegrität sowie die Fehlervermeidung bei konkurrierenden Zugriffen.

Heute sind relationale bzw. objektorientierte Datenbanksysteme am meisten im Gebrauch. Relationale Datenbanksysteme bieten eine logische Sicht auf Daten, denen das relationale Modell zugrunde liegt. Objektorientierte Datenbanksysteme ermöglichen dagegen eine logische Sicht auf Daten, denen das Objektmodell zugrunde liegt[1].

.

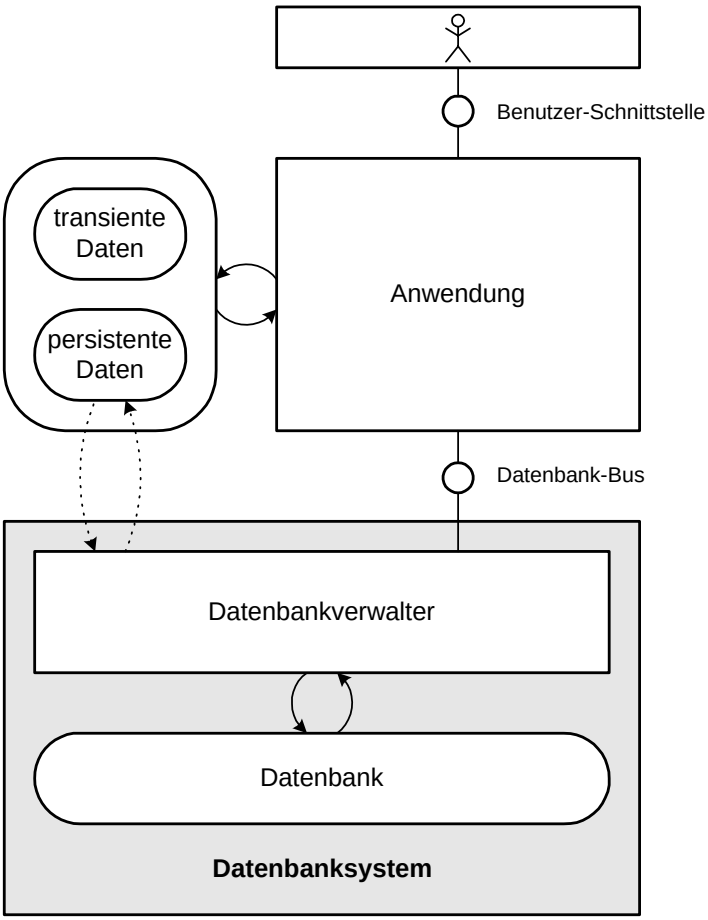


Bild 26: Aufbaubild einer Anwendung mit Datenbanksystem

Um Daten einer Anwendung abspeichern zu können, muß der Datenbankverwalter Informationen über die Struktur der abzulegenden Daten besitzen. Diese Information bezeichnet man als ***Datenbank-Schema***.

Das Schema einer relationalen Datenbank äußert sich in der Definition von Tabellen und Beziehungen zwischen den Tabellen. Bei objektorientierten Datenbanken zeigt sich dies in der Klassenbeschreibung aller ablegbaren Objekte.

Jede dieser Datenbanktechnologien kann für gewisse Anwendungsgebiete Vor- bzw. Nachteile bieten, auf die im folgenden kurz eingegangen wird.

1. Eine ausführliche Beschreibung dieser Modelle findet man in [Date 90] und [Lausen 96].

Relationale Datenbanksysteme

Vorteile:

- Performanter Suchmechanismus bei großen Datenmengen.
- Einfache Informationsdarstellung durch Tabellen.

Nachteile:

- Schwere Änderbarkeit des Schemas.
- Schlechte Einbettung der Datenbankanfragesprache (z.B. SQL) in die Programmiersprache der Anwendung.

Objektorientierte Datenbanksysteme

Vorteile:

- Schema an Anwendung angelehnt.
- Leichte Änderbarkeit des Schemas.

Nachteil:

- Suchanfragen sind in der Regel nicht sehr effizient.

2. Schnittstellen

Der Aufbau aus Bild 26, bei dem eine Anwendung die Dienste eines Datenbanksystems nutzt, hat den Nachteil, daß ein großer Teil der Anwendung abhängig von den genutzten Diensten ist. Soll nun das Datenbanksystem, auf das die Anwendung aufsetzt, ausgetauscht werden, so ist ein hoher Aufwand nötig, um die Anwendung daran anzupassen. Gründe für einen Austausch gibt es viele. Beispielsweise kann ein Kunde die Verwendung eines bestimmten Datenbanksystems zur Speicherung der persistenten Daten verlangen, da ihm nur für dieses geschulte Mitarbeiter zur Verfügung stehen.

Um zu vermeiden, daß große Teile der Anwendung abhängig von den angebotenen Diensten des Datenbanksystems sind, ist eine klare Trennung des Systems in verschiedene Komponenten nötig. In Bild 27 ist die Anwendung in vier Komponenten unterteilt dargestellt.

Das Basissystem beinhaltet die eigentliche Logik des Gesamtsystems. Die Tools greifen auf die Funktionalität des Basissystems über den Tool-Bus zu. Dabei stellen sie eine dem Benutzer angepaßte Interaktionsschnittstelle zur Verfügung, die meist nur eine Teilmenge der Funktionalität des Basissystems nutzt.

Das Basissystem kann beispielsweise das Erzeugen und das Löschen von flächig-statischen Kommentaren ermöglichen. Ein Tool kann dem Benutzer die Erstellung von Kommentaren mit reinem Text ermöglichen; ein anderes Tool könnte hingegen die Erstellung von Kommentaren mit Text und Grafiken zulassen. Daran erkennt man, daß die Menge der genutzten Funktionalität den Bedürfnissen der Benutzer angepaßt ist.

Der Logische-Service stellt dem Tool-Entwickler eine an die Semantik der Anwendung orientierte Schnittstelle zur Verfügung. Hierbei bleiben sowohl Dienste des Berechtigungsservice als auch Dienste des Persistenz-Service dem Tool-Entwickler verborgen. Über den System-Bus kann der Logische-Service bei Bedarf diese Systemdienste in Anspruch nehmen.

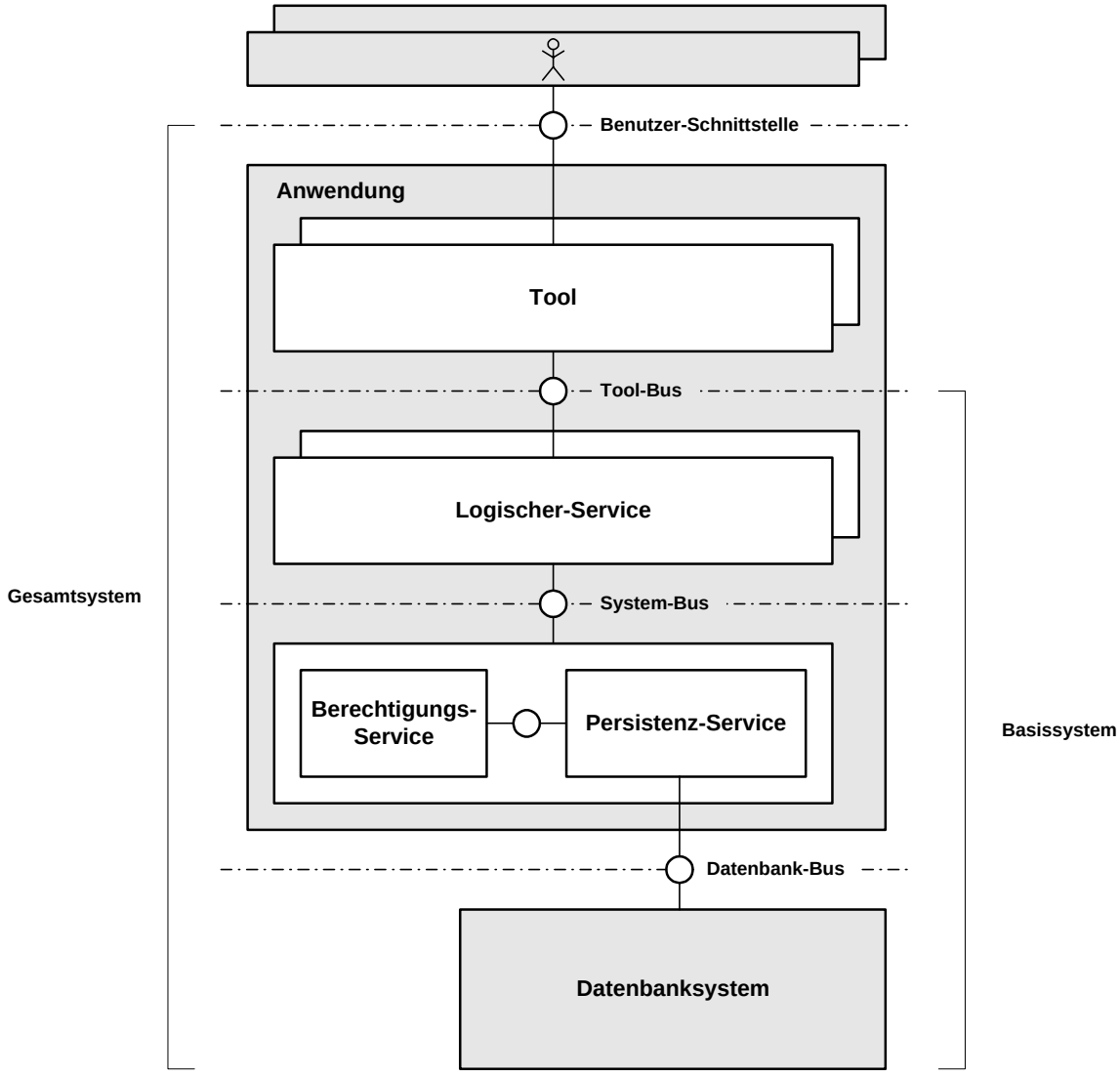


Bild 27: Gesamtsystem in Komponentendarstellung

3. Basissystem

Im folgenden soll die Aufmerksamkeit auf das Basissystem gerichtet werden, das in Bild 28 dargestellt ist. Die anwendungsspezifische Funktionalität des Basissystems wird durch die Logischen Elemente (LE) und die Persistenz Elemente (PE) bestimmt. Logische und Persistenz Elemente können zur Laufzeit des Basissystems entstehen und verschwinden. Die Verwaltung dieser Strukturvarianz übernimmt im Falle der Logischen Elemente die LE-Verwaltung und im Falle der Persistenz Elemente die PE-Verwaltung. Die Kommunikation der Elemente untereinander ist ebenfalls über die Verwaltungen möglich. Die Verhaltensbeschreibung und sowie der Bauplan der Logischen und Persistenz Elemente ist in den Elementtypdefinitionen festgelegt. Auf diese greift die Verwaltung zur Erzeugung der Logischen bzw. Persistenz Elemente zu.

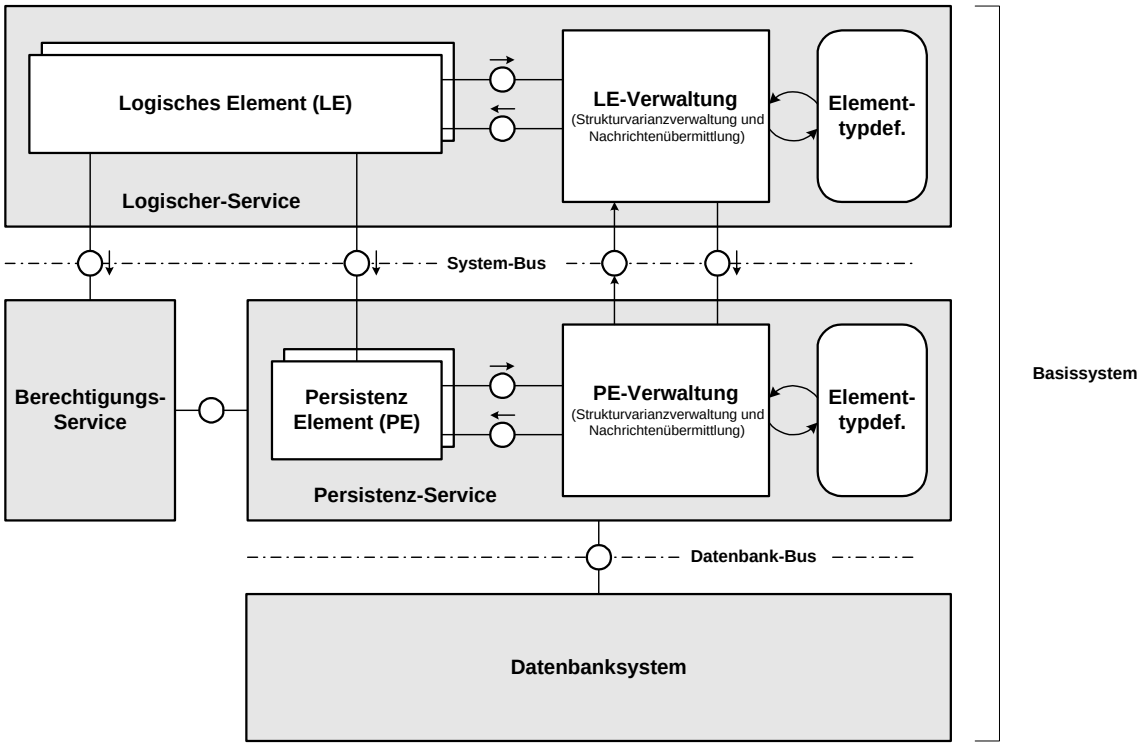


Bild 28: Basissystem

Persistente Elemente sind aktive Stellvertreter der in der Datenbank dauerhaft gespeicherten Objekte. Sie beinhalten persistente Daten, deren Struktur in Kapitel zwei erläutert wurde.

Logische Elemente sind die Elemente, über die die anwendungsspezifische Funktionalität des Basissystems angeboten wird. Logische Elemente können sowohl transient als auch persistent sein. Bei transienten Logischen Elementen ist der Zustand nach deren Erzeugung immer gleich (vorausgesetzt die Elemente sind Elemente des gleichen Typs). Persistente Logische Elemente übernehmen nach deren Erzeugung immer den Zustand des Persistenz Elements, deren Stellvertreter sie sind. Wenn im folgenden von Logischen Elementen die Rede ist, dann sind damit immer persistente Logische Elemente gemeint.

Für ein persistentes Element kann es mehrere Logische Elemente geben, die einen benutzerspezifischen Zugriff auf die vom Persistenz Element verwalteten Daten ermöglichen. Um dies zu gewährleiten, kommunizieren die Logischen Elemente über den Systembus mit dem Berechtigungsservice.

Da Logische Elemente nebenläufig agieren, gibt es Mechanismen, um Fehler zu vermeiden, die aufgrund von konkurrierenden Zugriffen auf Persistenz Elemente entstehen können. Aus diesem Grund bietet die LE-Verwaltung und die PE-Verwaltung erweiterte Dienste an.

Um diese Dienste besser verstehen zu können, wird zuerst der Mechanismus zur Aktualisierung der Logischen Elemente beschrieben. Bei einer Änderung eines Persistenz Elements sollen dessen Stellvertreter davon in Kenntnis gesetzt werden. Der Ablauf bei einer Änderung ist in Bild 29 an einem Beispiel erläutert.

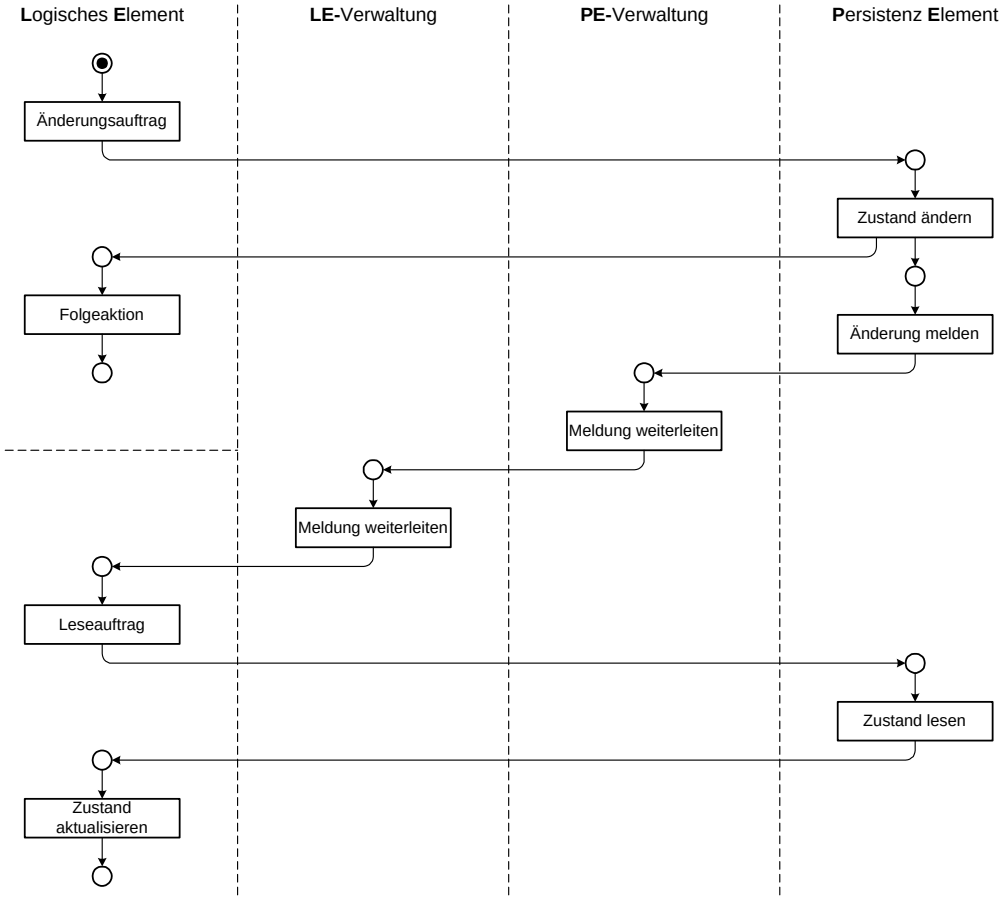


Bild 29: Allgemeiner Ablauf bei Zustandsänderung eines Persistenz Elements

Will ein Logisches Element seinen aktuellen Zustand dauerhaft speichern, gibt es seinem zugehörigen Persistenz Element den entsprechenden Auftrag. Das Persistenz Element übernimmt den Zustand des auftraggebenden Logischen Elements und meldet nach Beendigung des Auftrags dem auftraggebenden Element den Erfolg der Zustandsänderung zurück. Außerdem meldet es der PE-Verwaltung, daß sein Zustand verändert wurde. Die PE-Verwaltung leitet solche Änderungsmeldungen an die LE-Verwaltung des Logischen-Service weiter.

Die LE-Verwaltung besitzt eine sogenannte Subskriptionsliste, in der eingetragen wird, welche Logischen Elemente bei welcher Meldung der PE-Verwaltung benachrichtigt werden müssen. Im gezeigten Ablaufbild ist das nur ein Logisches Element. Dieses erhält eine Änderungsmeldung und weiß somit, daß der ihm bekannte Zustand keine Gültigkeit mehr besitzt. Um seinen Zustand aktualisieren zu können, fordert das Logische Element Daten des Persistenz Elements an.[1]

3.1 Konkurrierende Zugriffe

Es gibt zwei grundlegende Dienste, die zur Vermeidung von Fehlern bei konkurrierenden Zugriffen dienen.

3.1.1 Sperren

Ein Logisches Element kann verhindern, daß Schreib- bzw. Leseaufträge anderer Logischer Elemente an ein Persistenz Element abgegeben werden, indem er eine Sperre darauf beantragt.

Es können einzelne oder auch eine Menge von Persistenz Elementen gesperrt werden. Um ein Persistenz Element zu sperren muß ein Logisches Element die Sperren bei der Sperrverwaltung beantragen. Diese ist, wie in Bild 30 zu sehen, Teil der PE-Verwaltung. Deshalb muß der Auftrag von der LE-Verwaltung an die Sperrverwaltung weitergeleitet werden.

Die Sperrverwaltung prüft, ob der Sperrauftrag ausführbar ist. Ist dies der Fall, teilt sie den entsprechenden Persistenz Elementen mit, daß sie gesperrt werden. Nachdem die Sperrverwaltung die Bestätigung aller Persistenz Elemente erhalten hat, vermerkt sie diese als gesperrt.

Alle Logischen Elemente, die Stellvertreter der gesperrten Persistenz Elemente sind, werden durch den bereits besprochenen Aktualisierungsmechanismus von der Sperrung in Kenntnis gesetzt.

1. Eine ähnliche Vorgehensweise wird auch in [Pattern 95] Seite 293 ff. beschrieben.

3.1.2 Transaktionen

Eine Transaktion ist eine Zusammenfassung von Operationen, die wie eine atomare Operation behandelt wird. Das bedeutet, daß entweder alle Operationen oder keine ausgeführt werden. Dies nennt man die Atomarität einer Transaktion. Durch die Atomarität kann die Konsistenz des Systems gewährleistet werden.

Transaktionen sind in der vorgestellten Architektur auf drei Ebenen definiert:

- Logischer-Service
- Persistenz-Service
- Datenbanksystem

Logischer-Service

Im Logischen-Service ist häufig zu Beginn der Transaktion nicht bekannt, welche Daten geändert werden, da diese erst im Laufe einer Reihe von Benutzerinteraktionen festgestellt werden können. Das bedeutet, daß ein Benutzer eine Folge von Aktionen ausführen will, die nur in ihrer Gesamtheit gültig sein sollen, jedoch nicht zum Beginn der Interaktion feststehen. Ein typisches Beispiel ist die Buchung mehrerer Flüge in einem Flugbuchungssystem. Am Anfang des Buchungsvorgangs steht die Route noch nicht fest. Die Route wird erst im Laufe des Buchungsvorgangs festgelegt. Trotzdem soll die Buchung als Ganzes durchgeführt werden, wie es dem Wesen einer Transaktion entspricht.

Um diese Anforderung zu erfüllen, ist es notwendig den Logischen-Service auf festgelegte Zustände zurücksetzen zu können. Daher wurde hier die Möglichkeit geschaffen, *geschachtelte Transaktionen*[1] auszuführen. Hierbei können Transaktionen innerhalb anderer Transaktionen vorkommen, wobei der Zustand zum Anfangszeitpunkt der geschachtelten Transaktion gespeichert wird. Auf diese gespeicherten Zustände kann die im Rahmen der Gesamttransaktion geänderte Objektmenge jederzeit zurückgesetzt werden.

Persistenz Service / Datenbanksystem

Bei einer Transaktion des Persistenz Service bzw. des Datenbanksystems handelt es sich um Transaktionen, bei denen zu Beginn schon feststeht, welche Daten geändert werden. Aus diesem Grund ist es dort nicht nötig, geschachtelte Transaktionen wie bei dem Logischen Service anzubieten.

Auf die Implementierung und die weiteren Konzepte der Transaktionen der verschiedenen Ebenen kann im folgenden nicht weiter eingegangenen werden, da diese zum Ende der Arbeit noch nicht ausgearbeitet waren.

1. engl. nested transactions

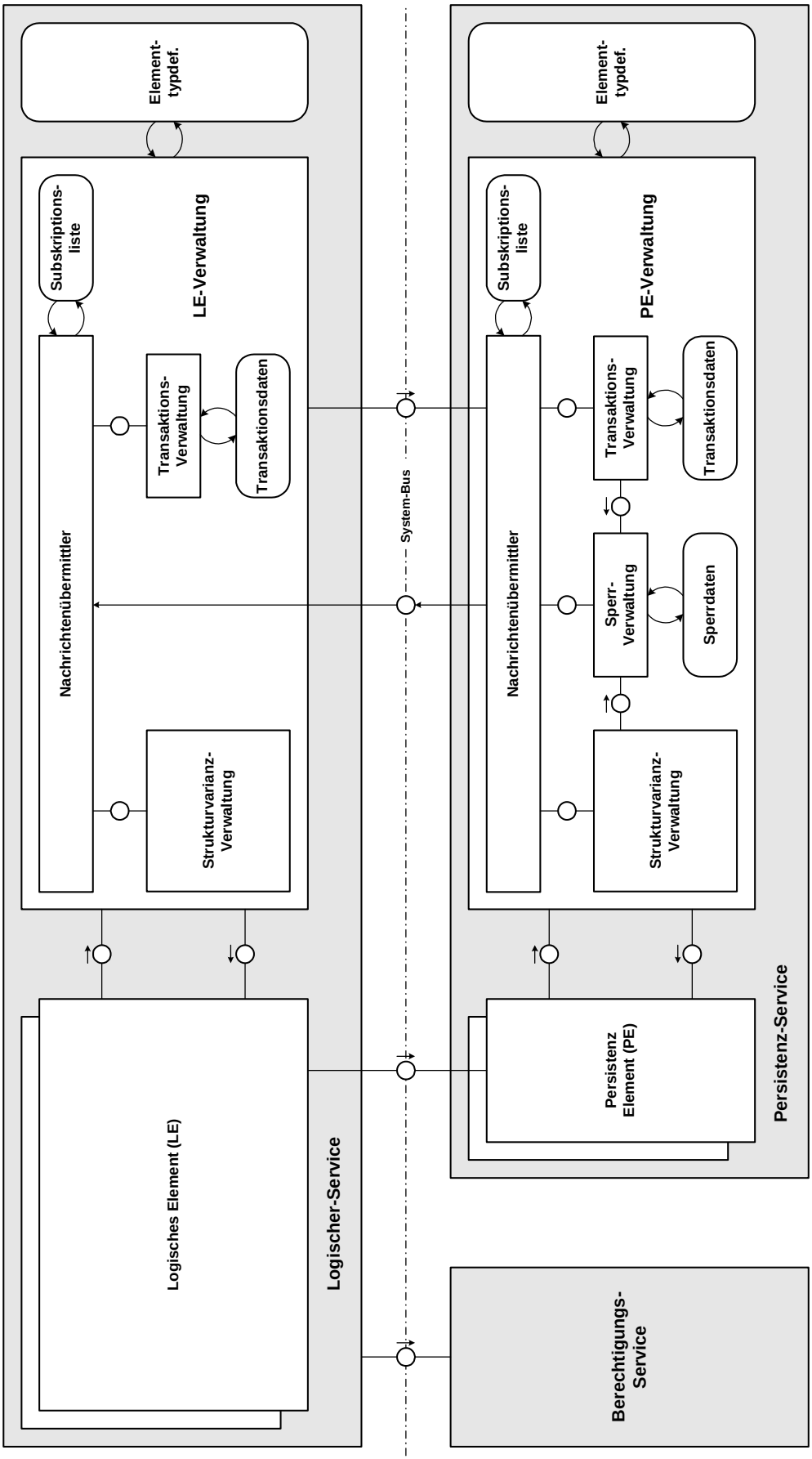


Bild 30: Verfeinerte Betrachtung eines Teils des Basissytems

IV Literaturverzeichnis

[ABAP 96] **ABAP\4 Development Workbench**
SAP, 1996

[Brand 97] Christian Brand
Spezifikation und Entwurf der Navigationskomponente für ein "Corporate Memory" und Implementierung eines Demonstrationsprototyps
Diplomarbeit am Lehrstuhl für Digitale Systeme, Universität Kaiserslautern, 1997

[Gröne 96] Bernhard Gröne
Bereitstellung einer Laborumgebung und Untersuchung objektorientierter Datenbanktechnologien
Diplomarbeit am Lehrstuhl für Digitale Systeme, Universität Kaiserslautern, 1996

[Lausen 96] Georg Lausen; Gottfried Vossen
Objektorientierte Datenbanken: Modelle und Sprachen
Oldenbourg-Verlag, 1996

[ODS 96] R.G.G. Cattell
The Object Database Standard: ODMG-93
Morgan Kausmann Publishers, Inc.

[Wendt 91] Siegfried Wendt
Nichtphysikalische Grundlagen der Informationstechnik
Springer, 1991

[Wiegert 95] Oliver Wiegert
Änderbarkeit durch Objektorientierung
Vieweg, 1995

[Date 90] C.J. Date
An Intoduction to Database Systems
Addison-Wesley, 1990

[Lock 87] P.C. Lockemann, J.W. Schmidt
Datenbank-Handbuch
Springer, 1987

[Pattern 95] E. Gamma, R. Helm, R. Johnson, J. Vlissides
Design Patterns:
Elements of Reusable Object-Oriented Software
Addison-Wesley, 1995