Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Diplomarbeit

UbiMex:
Ein persönliches Wissensmanagementsystem für Ubiquitous Computing

vorgelegt von
Michael Knop
am 4. Juni 2004

Studiengang Softwaretechnik

Betreuender Prüfer: Prof. Dr. Kai von Luck
Zweitgutachter: Prof. Dr.-Ing. Martin Hübner

Fachbereich Elektrotechnik und Informatik
Department of Electrical Engineering and Computer Science

UbiMex: Ein persönliches Wissensmanagementsystem für Ubiquitous Computing

Stichworte persönliches Wissensmanagementsystem, UbiMex, Ubiquitous Computing

Zusammenfassung

In der Dienstleistungsgesellschaft, auf dem Weg in die Wissensgesellschaft, nimmt die Bedeutung von Wissen immer mehr zu. Gleichzeitig ist ein Trend zu immer kürzeren Beschäftigungsverhältnissen zu erkennen. Der Einzelne wird gezwungenermaßen immer mehr zum Manager seiner selbst. Daraus ergibt sich die Notwendigkeit, dass das persönliche Kapital – das Wissen – professionell abgelegt und verwaltet wird. Diese Tatsache wurde zum Ausgangspunkt genommen, um ein Konzept für ein persönliches Wissensmanagementsystem zu erarbeiten. Die Zielsetzung lautet ein Instrument zu schaffen, das es dem Nutzer gestattet, jederzeit und an jedem Ort auf sein persönliches Wissen zugreifen zu können. Gesucht wird eine Ablagestrategie, die mit minimalem Aufwand und ohne Vorarbeiten zu handhaben ist. Zur Zielerreichung wurde der Terminkalender als Startpunkt gewählt. Das Ergebnis ist der Prototyp eines persönlichen Wissensmanagementsystems, dass mit Standardkomponenten umgesetzt wurde. Bei diesen Standardkomponenten handelt es sich um frei verfügbare Produkte.

UbiMex: A Personal Knowledge Management System for Ubiquitous Computing

Keywords personal knowledge management system, UbiMex, Ubiquitous Computing

Abstract

Within service society, on its way to knowledge society, the importance of knowledge is growing constantly. At the same time there is a tendency to shorter employments. The individual is forced to be the manager of himself. There is a necessity to manage ones only asset – personal knowledge. This fact is the base for building a concept of a personal knowledge management system. The aim was to create an instrument, which allows access to personal knowledge anywhere and anyplace. A search for filing strategy with minimal effort and without the need of any preparatory work. To realize this goal the calendar was chosen as a starting point. The result is the prototype of a personal knowledge management system. This prototype was implemented with open source building blocks.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1. Einleitung und Motivation

In der Arbeitswelt vollzieht sich derzeit ein Wandel. Immer weniger Arbeitnehmer sind von der Berufsausbildung bis zur Rente bei derselben Firma beschäftigt. Lebenslange Beschäftigung wird zunehmend durch Projektarbeit abgelöst. Immer mehr Menschen werden sich im Beruf schon bald als eigenverantwortlich verstehen (müssen). Jedes Individuum ist für sich selbst verantwortlich und gestaltet das eigene Berufsleben individuell wie ein Unternehmer als so genannte „Ich-AG“. Obwohl sich die Diskussion über die Ich-AG merklich beruhigt hat und – um es mit den Worten von Konrad Adenauer zu sagen – bereits eine neue Sau durchs Dorf getrieben wird, besteht der Trend fort. Unabhängig davon, wie er gerade genannt wird. Dieser Trend ist bereits heute zum Beispiel in der Medienbranche zu erkennen. Dort bestehen die Unternehmen aus einer kleinen, fest angestellten Kernmannschaft. Bei Bedarf wird die Kernmannschaft durch Freiberufler, den Ich-AGs, ergänzt (Englert 2001).

In der Diskussion wird unter dem Schlagwort Ich-AG vornehmlich die Selbstvermarktung gesehen. Es werden also Fragen gestellt, wie: „Was kann ich besonders gut und wie kann ich dies am besten nach außen verkaufen?“ Unter der Bezeichnung Ich-AG soll hier jedoch vielmehr das Management des eigenen Marktwertes verstanden werden. Denn bisher haben sich die Personalabteilungen der Unternehmen darum gekümmert, dass die Leistungsfähigkeit jedes Einzelnen durch Weiterbildung erhalten bleibt. In einer Ich-AG muss nun selbst darauf geachtet werden, das eigene Wissen auf dem Laufenden zu halten. Lebenslanges Lernen wird zur existenzsichernden Pflicht, da nur auf diese Weise der eigene Marktwert erhalten und gesteigert werden kann.

„Wissen wird neben Kapital und Arbeitskraft zum immer entscheidenderen Faktor der betrieblichen Leistung“[1] (Drucker 1993, Seite 20). Sveiby spricht in diesem Zusammenhang von „Wissenskapital“ (Sveiby 1997). Speziell in der tertiären Gesellschaft, in dessen Vorstadium wir uns befinden, ist Wissen von immenser Bedeutung. In Agrar- oder Industriegesellschaften spielen Fähigkeiten und Fertigkeiten eine zusätzliche Rolle. Diese wird in der Dienstleistungsgesellschaft Schritt für Schritt durch Wissen in den Hintergrund gedrängt – auf dem Weg in die Wissensgesellschaft (vgl. Kubicek 1985).

Sogar die Bilanzierungsrichtlinien berücksichtigen inzwischen das Wissenskapital. So empfiehlt das Deutsche Rechnungslegungs Standards Committee (DRSC 2002) eine Berichter-

[1] "Knowledge is now fast becoming the sole factor of production, sideling both capital and labor."

stattung über das intellektuelle Kapital einer Firma. Was auf Unternehmen im herkömmlichen Sinne zutrifft, gilt im besonderen Maße auch für die Ich-AGs. Im Gegensatz zu den Unternehmen besitzen Ich-AGs weit weniger materielle Unternehmenswerte. Der Marktwert einer Ich-AG besteht zum großen Teil aus angesammeltem Wissen. Einmal erworbenes Wissen muss erhalten und gesichert werden. Man könnte nun geneigt sein, in Zeiten von Windows XP seine alten Unterlagen über das alte Betriebssystem Windows 3.1 zu löschen. Was aber, wenn dann doch noch ein Kunde Probleme mit einem Altsystem hat, das noch unter Windows 3.1 läuft? Digitale Speichermedien sind inzwischen derartig günstig geworden, dass es sinnvoller erscheint, nichts zu löschen, als das Risiko einzugehen, etwas zu löschen, das später noch benötigt wird. Das abgespeicherte Wissen muss nicht nur wiedergefunden werden, es muss zusätzlich auch stets auf dem aktuellen Stand gehalten werden.

Dies gilt insbesondere für so genannte „Wissensarbeiter“. Hierbei handelt es sich um hoch qualifizierte und gut ausgebildete Spezialisten, deren Arbeit größtenteils darin besteht, Informationen in Wissen umzuwandeln und dieses Wissen anzuwenden (Davis 2002, Seite 68). Im Folgenden sind mit der Bezeichnung Ich-AG auf sich allein gestellte Wissensarbeiter gemeint, sozusagen „Wissens-Ich-AGs“.

Die Ich-AG erwirtschaftet nur durch das Anwenden des angesammelten Wissens einen Mehrwert, jedoch nicht durch die Suche nach benötigten Informationen. Zum Beispiel kann die Suche nach einer bestimmten Funktion in einer umfangreichen Klassenbibliothek leicht mehrere Stunden in Anspruch nehmen. Somit entspricht die Information über diese bestimmte Funktion dem Gegenwert von mehreren Arbeitsstunden. Auf diese Information sollte beim nächsten Mal, wenn sie benötigt wird, schnell zugegriffen werden können.

Da Wissenserhalt und Weiterbildung zusätzlich zur eigentlichen Arbeit erledigt werden müssen, können nur entsprechende Hilfsmittel den erforderlichen Arbeitsaufwand im Rahmen halten. Diese Hilfsmittel können bei der Verwaltung des erworbenen Wissens und beim Zugriff auf das gespeicherte Wissen behilflich sein. Sie verwalten somit das persönliche Eigentum, das Wissen, einer Ich-AG: ein persönliches Wissensmanagementsystem. Diesem System wurde der Name „UbiMex“ gegeben. Er leitet sich von den beiden Begriffen „Ubi“ und „Mex“ ab. „Ubi“ von „Ubiquitous Computing“[2], Mark Weisers (1991) Vision von der dritten Computergeneration. Nach dieser Vision sollen die Computer in die Welt, anstatt die Welt im Computer nachzubilden. „Mex“ von „Memex“, hierbei handelt es sich um die Idee von Vannevar Bush (1945), ein System zur Erweiterung des menschlichen Gedächtnisses zu erstellen.

[2] ubique: lat. überall

1.1. Übersicht

Im Rahmen dieser Arbeit wird ein System zur strukturierten Ablage des persönlichen Wissens entworfen. Es wird eine Ablagestrategie gesucht, die es ermöglicht, Informationen bei minimalem Aufwand und ohne Vorarbeiten abzulegen. Die abgelegten Informationen müssen derartig strukturiert sein, dass ein leichtes Wiederfinden gegeben ist. Des Weiteren wird der Aspekt berücksichtigt, Informationen jederzeit und an jedem Ort nutzbar zu machen. Denn das persönliche Wissen kann nur dann Gewinn bringend eingesetzt werden, wenn es auch dort zur Verfügung steht, wo es gebraucht wird. Hierfür wird zum einen die Möglichkeit vorgesehen, unterschiedliche Datenformate zu verarbeiten. Zum anderen wird der Zugriff auf die Daten von unterschiedlichen Orten behandelt.

Aufbau

In Kapitel 2 (**Grundlagen**) wird das domänenspezifische Basiswissen dargelegt, soweit es zum Verständnis der Arbeit notwendig ist. Die informatikspezifischen Fachbegriffe werden jeweils im entsprechenden Kontext, unmittelbar vor ihrer Verwendung, eingeführt. Somit finden sich in Kapitel 2 Erörterungen auf Fragen bezüglich des Wesens vom Wissen allgemein und der momentanen Art der Verwaltung von digital vorliegendem Wissen. Welche Methoden existieren, um Informationen auch wiederzufinden? Die Vision von der allgegenwärtigen Datenverarbeitung und den damit einhergehenden allgegenwärtigen Informationen werden beschrieben. Weiterhin findet eine Abgrenzung zu bereits bestehenden Lösungen statt.

Nach der Betrachtung der Grundlagen werden in Kapitel 3 (**Analyse der Aufgabenstellung**) die Anforderungen an das zu erstellende System erarbeitet. In Form von Anwendungsfällen werden hierbei die mit dem System durchzuführenden Aufgaben beschrieben.

Auf Basis der Anforderungen und Anwendungsfälle wird in Kapitel 4 (**Entwurf einer möglichen Lösung**) eine Lösung erarbeitet. Hierbei bilden die Objekte aus der realen Welt den Ausgangspunkt. Aus einem realen Objekt wird in der Analyse durch Modellbildung und geeignete Abstraktion ein Objekt des objektorientierten Modells. Ziel der Analyse ist es, das zu realisierende Problem zu verstehen und in einem Objektmodell zu beschreiben. Dieses Modell soll die wesentliche Struktur und Semantik des Problems, allerdings noch keine technische Lösung des Problems erörtern. In der Analyse wird der wichtigste Bestandteil des Systems, das Informationsobjekt, herausgearbeitet. Im Anschluss werden die unterschiedlichen Verfahren zur Ablage von Informationen diskutiert, um dann das passend erscheinende Verfahren auszuwählen. Danach wird eine Systemarchitektur für das in der Analyse spezifizierte System entwickelt. Dieses System wendet das ausgewählte Verfahren zur Ablage von Informationen an. Das im Rahmen der Analyse erstellte Modell eines Informationsobjekts wird nun stark vereinfacht. Der Grund hierfür liegt darin, dass die exakte Nachbildung

der Wirklichkeit die Umsetzung des Systems zwar komplizierter machen würde, die genaue Abbildung jedoch keinen entscheidenden Mehrwert bringen würde.

Der in Kapitel 4 erarbeitete Entwurf wird im Kapitel 5 (**Realisierung**) ob seiner Realisierbarkeit hin untersucht. Dazu wird die erarbeitete Systemarchitektur prototypisch umgesetzt. Das System wird nicht von Grund auf neu erstellt. Vielmehr wird auf vorhandene Komponenten zurückgegriffen. Das Rad muss auf diese Art und Weise nicht neu erfunden werden und es kann auf bereits bewährte Technik aufgebaut werden. Entsprechend werden die benötigten Komponenten ausgewählt.

Eine kritische Würdigung des hier vorgestellten Ansatzes und ein Ausblick auf mögliche Weiterentwicklungen findet sich als Abschluss in Kapitel 6 (**Fazit und Ausblick**).

2. Grundlagen

Im folgenden Kapitel werden die Grundbegriffe aus dem Anwendungskontext erläutert. „Wissen“ stellt den Kernbegriff dieser Arbeit dar. Daher soll im Zusammenhang mit der Verwaltung von Wissen auch der Begriff selbst kurz betrachtet werden. Es existieren unterschiedliche Ausrichtungen im Bereich der Verwaltung von Wissen. Aus diesem Grund wird das hier zu entwickelnde System mit den grundsätzlichen Richtungen verglichen. Nachdem sich mit Wissen allgemein und dem Management von digital vorliegendem Wissen im Speziellen beschäftigt wurde, werden unterschiedliche Möglichkeiten der Ablage von Wissen betrachtet. Das Wissen muss derartig abgelegt werden, dass später benötigte Informationen auch wiedergefunden werden können. Schließlich wird die Vision von der allgegenwärtigen Datenverarbeitung und den damit einhergehenden allgegenwärtigen Informationen im Bezug zu Wissensmanagementsystemen betrachtet. Den Abschluss des Kapitels bildet ein Überblick von Projekten, die ähnlich gelagerte Probleme behandeln.

2.1. Management von digital vorliegendem Wissen

Eine derartig wertvolle Ressource wie Wissen, muss auch ihrer Bedeutung entsprechend behandelt werden. Sie bedarf eines qualifizierten Managements. Da die Ich-AG zumeist als Einzelkämpfer agiert, werden Hilfsmittel zum Management der wichtigsten Ressource benötigt.

Im Zusammenhang mit Management und Erschließbarmachung von digital vorliegendem Wissen werden immer wieder die Begriffe „Wissensmanagement“ und „Content Management“ verwendet. Die Begriffe sind beide nicht eindeutig definiert. Es stellt sich die Frage, wie sich Wissensmanagement und Content Management voneinander unterscheiden.

Doch zunächst soll erstmal näher betrachtet werden, was überhaupt unter dem Begriff „Wissen“ verstanden wird.

Was ist Wissen?

> Wo ist die Weisheit, die wir im Wissen verloren haben?
> Wo ist das Wissen, dass wir in den Informationen verloren haben?[1]

(T.S. Eliot, “Choruses from the Rock“, 1934)

„Wissen könnte als Information definiert werden, die mit Erfahrung, Kontext, Interpretation und Reflexion kombiniert worden ist“[2] (Davenport u. a. 1997, Seite 1). Cleveland (1982) bedient sich zur Definition des Begriffs „Wissen“ bei dem Schriftsteller T.S. Eliot und der dort formulierten Hierarchie. Ackoff (1989) formuliert eine ähnliche Hierarchie aus Daten, Information, Wissen und Weisheit. Daten bilden die unterste Ebene. Sie setzen sich aus unverarbeiteten Fakten zusammen. Verknüpft mit dem Kontext, in dem sie stehen, werden Daten zu Informationen verarbeitet. Informationen liefern Antworten auf Wer-, Was-, Wo- und Wann-Fragen. Wissen ist die Anwendung dieser Daten und Information, und liefert zusätzlich Antworten auf Wie-Fragen.

„Wissen befindet sich in der Person und nicht in der Ansammlung von Informationen. Es ist die Art, wie die Person auf eine Ansammlung von Informationen, die von Bedeutung sind, reagiert.“[3] (Churchman 1971, Seite 10). Auch nach der Logik von Michael Polanyi ist Wissen unmittelbar an Personen gebunden („Personal Knowledge“). Es entsteht in Personen und wird von Personen angewendet. Es lässt sich dementsprechend auch nicht von ihnen ablösen (Polanyi 1966). Wissen ist entweder als so genanntes „Implizites Wissen“ an Personen gebunden oder hat seinen Ursprung darin. Im Gegensatz dazu wird das Wissen, welches in Worte gefasst werden kann und somit auch digital speicherbar ist, als „Explizites Wissen“ bezeichnet. Über das explizite Wissen – in Form einer Ansammlung von relevanten Informationen – kann versucht werden, den Zugriff auf das persönliche Wissen zu vereinfachen und Hilfsmittel bereitzustellen, um das persönliche Wissen nicht zu vergessen.

Content Management

In der Literatur wird unter Content Management die Erstellung und Verwaltung von digitalen Inhalten verstanden (vgl. z. B. Schuster und Wilhelm 2000). Ein besonderes Augenmerk liegt hier auf der Konsistenz und Erschließbarkeit der Inhalte. In der Regel wird der Begriff Content Management im Zusammenhang mit Inhalten verwendet, die im weltweiten Internet oder

[1] Where is the wisdom we have lost in knowledge? Where is the knowledge we have lost in information?

[2] "Knowledge could be defined as information that has been combined with experience, context, interpretation, and reflection."

[3] "Knowledge resides in the user and not in the collection. It is how the user reacts to a collection of information that matters."

im internen Intranet veröffentlicht werden. Wesentliches Merkmal von Content Management Systemen ist die Trennung von Struktur, Inhalt und Präsentation (vgl. Abbildung 2.1).

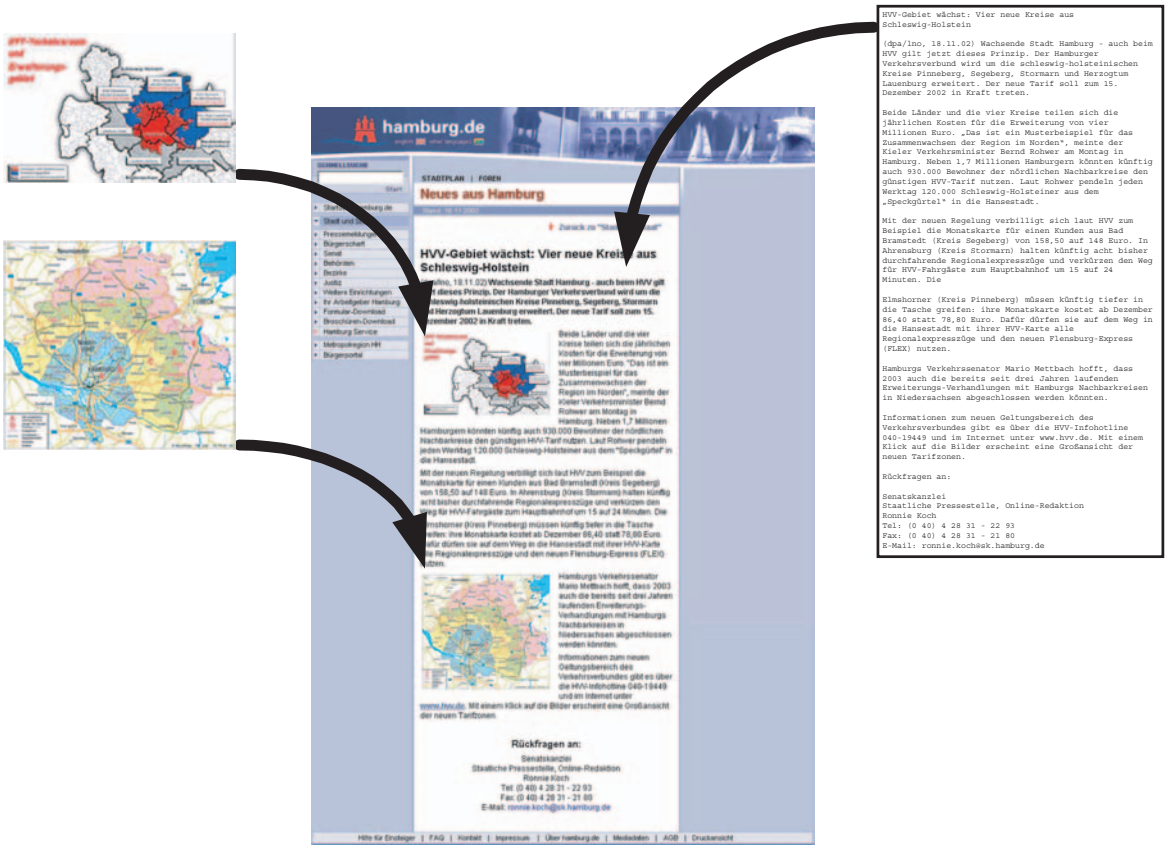

Abbildung 2.1.: Die Daten werden getrennt voneinander gespeichert und erst bei Bedarf zusammengesetzt.

Durch die Trennung von Inhalt und Präsentation kann die Ausgabe der Inhalte an die Fähigkeiten von verschiedenen Geräten angepasst werden, mit denen auf die Inhalte zugegriffen wird. Mögliche Ausgabeformate sind zum Beispiel HTML für die herkömmlichen Internetbrowser auf PCs, WML für mobile Geräte, PDF zum Drucken – um nur einige zu nennen. Während der Internetbrowser auf dem PC ein Dokument als HTML-Datei ausgeliefert bekommt, kann zum Ausdrucken der Datei eine PDF-Version erstellt werden. Wenn das gleiche Dokument von unterwegs mit einem Mobiltelefon angezeigt werden soll, wird eine WML-Datei ausgeliefert. Im Unterschied zur PDF-Version, die hochauflösende Grafiken enthält, wird das Dokument in der WML-Version ohne Grafiken, beziehungsweise mit weniger hochauflösenden Grafiken ausgeliefert. Die Ausgabe der Informationen erfolgt in einem Format, welches ganz auf das jeweilige Gerät abgestimmt ist.

Wissensmanagement

Im Vordergrund des Wissensmanagements steht der verbesserte Zugang zu Informationen und die Generierung von unternehmensspezifischem Wissen in einer Organisation (vgl. Davenport und Prusak 1997). Der Wunsch: „Wenn unsere Mitarbeiter nur wüssten, was unsere Mitarbeiter wissen. . . “, verdeutlicht die Intention des Wissensmanagements. Menschliches Wissen soll informationstechnisch verarbeitbar, auffindbar und transferierbar gemacht werden. Hierzu sollen die relevanten Informationen aller Individuen einer Organisation zentral abgelegt werden. Eine Organisation kann durch einen derart automatisierten Wissenstransfer Kosten einsparen. Ganz nebenbei wird die Einarbeitungszeit in die vom Wissensmanagementsystem verwalteten Themengebiete reduziert.

Wissens- versus Content Management

Beim Wissensmanagement ist ein entscheidender Punkt die automatisierte Weitergabe von Wissen zwischen Personen innerhalb einer Organisation. Es wird davon ausgegangen, dass das Wissen der Organisation gehört. Im Unterschied dazu gehört das Wissen einer Ich-AG nur einer Person. In Tabelle 2.1 sind die wesentlichen Unterschiede aufgeführt. Im Hinblick auf Ich-AGs kann der automatisierte Wissenstransfer vernachlässigt werden. Hier ist der manuelle Wissenstransfer entscheidender. So wird Wissen als Tauschobjekt zwischen Personen manuell ausgetauscht. Für den Einsatz in Ich-AGs ist die Strukturierung und Verwaltung von Informationen wichtig. Auch soll auf die vorhandenen Informationen überall und mit den verschiedensten Geräten zugegriffen werden können. Ein Hilfsmittel für Ich-AGs wäre somit zwischen den beiden Begriffen Wissensverwaltung und Content Management anzusiedeln.

Content Management	Wissensverwaltung
Verwaltung	Strukturierung von Inhalten
Darstellung	Wissenstransfer

Tabelle 2.1.: Gegenüberstellung von Content Management und Wissensmanagement

2.2. Wiederfinden von abgespeichertem Wissen

Beim Management des persönlichen Wissens ist der Transfer von Wissen zwischen einzelnen Individuen nicht so entscheidend, wie beim Wissensmanagement für Organisationen. Denn Wissen muss nicht zwischen verschiedenen Personen transferiert werden. Entscheidender ist, dass die archivierten Informationen auch wieder gefunden werden. Im folgenden

Abschnitt werden Möglichkeiten vorgestellt, mit denen der Zugriff auf gespeicherte Informationen organisiert werden kann.

Klassifikationssysteme

Klassifikationssysteme beruhen auf dem Prinzip der Klassenbildung. Es werden Begriffe zusammengefasst, die in mindestens einem klassenbildenden Merkmal übereinstimmen. Die einzelnen Klassen sollten das jeweilige Themengebiet vollständig abdecken und es sollte keine Überschneidung von Inhalten der unterschiedlichen Klassen geben. Bei der Suche nach Informationen, die nach einer derartigen Methode abgelegt sind, kann durch die Kategorien navigiert werden. Es kann vom Allgemeinen – Schritt für Schritt – zum Speziellen gelangt werden (Deduktion).

Die weltweit größte Vereinigung von Fachleuten und Wissenschaftlern der Informationstechnologie, die Association for Computing Machinery (ACM), hat zum Beispiel ihr Fachgebiet kategorisiert. Abbildung 2.2 zeigt einen kleinen Auszug aus dem Klassifikationssystem der ACM. Das erste Klassifikationssystem der ACM wurde 1964 veröffentlicht. Im Jahre 1982 wurde ein komplett neues System veröffentlicht. Auch dieses System wurde mehrfach angepasst (ACM 1998).

```
D.2 SOFTWARE ENGINEERING
# D.2.11 Software Architectures
    * Data abstraction
    * Domain-specific architectures
    * Information hiding
    * Languages
     (e.g., description, interconnection, definition)
    * Patterns
     (e.g., client/server, pipeline, blackboard)
```

Abbildung 2.2.: Auszug aus dem Klassifikationssystem der ACM

Die Erstellung eines Klassifikationssystems ist aufwendig und erfordert gute Kenntnisse in dem gesamten Themengebiet. Voraussetzung für die effiziente Nutzung ist die gute Kenntnis vom Aufbau des Systems. Vor allem muss vor der Verwendung eines Klassifikationssystems das Selbige modelliert werden.

Ontologien

Der Begriff „Ontologie“ stammt aus der Philosophie. Es werden dort Fragen bezüglich des Daseins und der Eigenschaften alles Existierenden behandelt. Aus Informatiksicht wird unter einer Ontologie die konzeptuelle Formalisierung von Wissensgebieten verstanden (Gruber 1993). Ein Teil der Realität wird auf eine Datenmenge abgebildet. Ontologien ordnen die relevanten Begriffe hierarchisch in Kategorien an. Diese Kategorien können mit anderen Kategorien über Relationen verknüpft oder mit Attributen näher beschrieben werden. Ziel ist es, ein Wissensgebiet für Maschinen verständlich zu machen.

Ontologien sind zurzeit zusammen mit „Semantischen Netzen“ in der Forschung ein stark bearbeitetes Gebiet (vgl. z. B. Visser 2004).

Neben den Verfahren, für die ein Modell oder eine Regelbasis aufgestellt werden muss, gibt es Verfahren, die ohne dies auskommen. Diese Verfahren verzichten zwar auf Vorarbeiten, der Aufwand während der Anwendung ist jedoch relativ hoch. Preece (1981) beschreibt beispielsweise das graphenbasierte Verfahren „Spreading Activation“, um in unstrukturierten Daten zu suchen. In dieser Arbeit wird jedoch davon ausgegangen, dass dieser Aufwand bei den eigenen Daten nicht nötig ist. Es werden nur Hilfen zum Wiederfinden benötigt. Das Wiederfinden oder die Erkennung von Ähnlichkeiten muss daher nicht automatisiert werden.

Syntaktische versus semantische Verweise

Allgemein wird zwischen syntaktischen und semantischen Verweisen unterschieden. Ein syntaktischer Verweis definiert eine Beziehung zwischen zwei Objekten, die nicht näher spezifiziert wird. Wobei die Beziehung selbst wertneutral hingenommen wird. Verweise in HTML-Dokumenten sind syntaktische Verweise. Ein Verweis in einem HTML-Dokument sieht wie folgt aus:

```
<a href="http://www.haw-hamburg.de/">HAW</a>
```

Der Text „HAW“ wird in diesem Beispiel mit dem Verweis auf die Internetseite `http://www.haw-hamburg.de/` verknüpft. Durch den Verweis ist eine Beziehung zwischen dem Text und der Internetseite gegeben. Aber was bedeutet der Verweis? Welche inhaltliche Beziehung besteht zwischen dem Verweis und dem Verweisziel? Diese Fragen bleiben bei syntaktischen Verweisen unbeantwortet.

Im Gegensatz zum bloßen In-Beziehung-Setzen mittels syntaktischer Verweise, werden semantische Verweise mit zusätzlichen Informationen angereichert. Ein semantischer Verweis für das obige Beispiel könnte wie folgt aussehen:

```
<a href="http://www.haw-hamburg.de/"
   role="Hochschule">HAW</a>
```

Zusätzlich zu der Spezifizierung des Verweises (role=“Hochschule“) wird eine Erklärung der Spezifizierung benötigt. Es muss erklärt werden, was „role“ beschreibt. So können die Informationen auch von Maschinen „verstanden“ werden. Semantische Verweise werden zum Beispiel im „Semantic Web“ verwendet (W3C 2001).

Kontext versus Kotext

In der Linguistik wird zwischen Ko- und Kontext unterschieden. Der Kontext bezeichnet dort die syntaktische Umgebung einer sprachlichen Einheit. Der „Kotext“ ist ein Kunstwort, welches den situationellen Zusammenhang bezeichnet (Catford 1969). Aus Sicht der Linguistik ist somit immer dann, wenn im Folgenden von „Kontext“ die Rede ist, eigentlich „Kotext“ gemeint.

Erweiterter Hypertext

Vannevar Bush hatte bereits 1945 die Grundlagen der assoziativen Verknüpfung von gespeicherten Informationen beschrieben (Bush 1945). Die Idee, Dokumente in einer nichtlinearen Form miteinander zu verbinden, wurde im Laufe der Zeit von verschiedenen Leuten umgesetzt. So realisierte der Erfinder der Computermaus, Douglas C. Engelbart, 1963 ein entsprechendes System. Die Bezeichnung „Hypertext“ wurde von Ted Nelson geprägt (Kuhlen 1991). Diese Idee wurde später von Tim Berners-Lee aufgegriffen, als er die Hypertext Markup Language (HTML) entwickelte (Berners-Lee 1989). Bei Hypertext können alle Dokumente miteinander verknüpft werden. Das von Vannevar Bush skizzierte System „Memex“ soll zusätzlich zur Möglichkeit der Verknüpfung von Dokumenten auch das Hinzufügen von zusätzlichen Informationen zu den Einträgen bieten. Im Gegensatz zu Hypertext sollen die Verknüpfungen bei Memex in beide Richtungen benutzbar sein. Ein Dokument „weiß“, welche Verweise auf das Dokument zeigen. So können Informationen in den Kontext abgespeichert werden, in dem sie ursprünglich aufgetreten sind. Zusammenhängende Informationen werden verknüpft. Zusätzlich können die Einträge mit weiteren Informationen versehen werden. Durch die Verknüpfung der Einträge untereinander und die Verknüpfung der Einträge mit dem Kontext, in dem sie aufgetreten sind, können „vergessene“[4] Teile durch Verfolgung der Verknüpfungen wieder gefunden werden.

[4] Da einmal gespeicherte Informationen nicht verschwinden, müsste hier korrekt eher von „verdrängten“ Informationen die Rede sein.

2.3. Allgegenwärtige Informationen

Wissen ist Information, die von Bedeutung ist. Diese Information ist nur relevant, wenn auch dort auf sie zugegriffen werden kann, wo sie benötigt wird. Ein persönliches Wissensmanagementsystem sollte demzufolge beim Auffinden von relevanten Informationen helfen und diese Informationen dort bereitstellen, wo sie benötigt werden. Die Bedeutung von allgegenwärtig verfügbaren Informationen wird immer größer. Nach einer Studie des Bundesministeriums für Wirtschaft und Technologie stellen Informationsdienste einen Kernbereich mobiler Multimediadienste dar (Beyer u. a. 2001, Seite 22). Nach dieser Studie ist der Informationssektor – langfristig betrachtet – der Markt der Zukunft.

Auf dem Weg zum Ubiquitous Computing

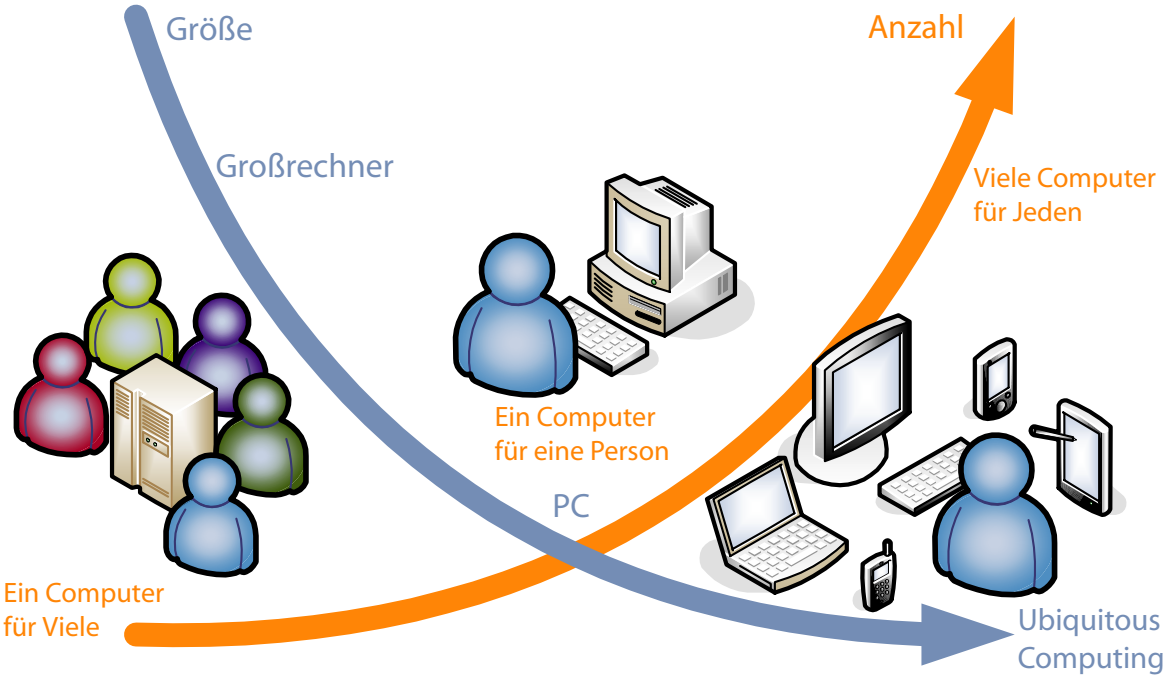

Abbildung 2.3.: Trends: Entwicklung vom Großrechner über den PC zur allgegenwärtigen Datenverarbeitung. Während die Rechner immer kleiner werden, nimmt die Anzahl an Rechnern, die die einzelne Person besitzt, zu.

Informationen allgegenwärtig anzubieten stellt einen Kernpunkt von Mark Weisers Vision vom „Ubiquitous Computing“[5] dar. Computer sind in dieser Vision zwar allgegenwärtig. Die

[5] ubique: lat. überall

Technik tritt aber in den Hintergrund und fungiert nur noch als reines Mittel zum Zweck (Weiser 1991). Auch der mit Ubiquitous Computing verwandte Begriff „Pervasive Computing“[6] wird über den allgegenwärtigen Zugriff auf relevante Informationen definiert. Die Bezeichnung „Pervasive Computing“ wurde durch die Firma IBM geprägt. IBM definiert Pervasive Computing wie folgt: „Einfacher Zugang, mittels einer neuen Geräteart, zu relevanten Informationen, mit der Möglichkeit, auf die Informationen reagieren zu können, wann und wo immer es nötigt ist.“[7] (nach Hansmann u. a. 2001, Seite 11).

Am Anfang der technischen Entwicklung der Computer standen die Großrechner (vgl. 2.3). Eine leistungsfähige und teure Maschine wurde von vielen Anwendern gemeinsam genutzt. Diese Ära wurde von den PCs abgelöst. Nun hatte jeder seinen „persönlichen“ Computer. Nach Weiser soll nun im nächsten Entwicklungsschritt jeder Benutzer viele verschiedene Computer einsetzen. Für jeden speziellen Anwendungsfall gibt es dann den dafür passenden Computer. Dadurch, dass die verschiedenen Geräte auf die Einsatzzwecke genau abgestimmt sind, kann die Bedienung intuitiver gestaltet werden. Die Technik tritt in den Hintergrund. Mark Weiser vergleicht dies mit der wohl ältesten Informationstechnologie, der Fähigkeit, die symbolische Repräsentation einer gesprochenen Sprache zu speichern. Zum Lesen und Schreiben wird keine bewusste Aufmerksamkeit benötigt. Das In-Den-Hintergrundtreten ist keine technische Konsequenz, vielmehr eine psychologische.

Mobile Computing

Derzeit werden drei separate Ausprägungen von Ubiquitous Computing anhand der Eigenschaften „mobil“ und „allgegenwärtig“ unterschieden (Lyytinen und Yoo 2002). Die Einteilung der Bereiche ist in Abbildung 2.4 dargestellt. Beim „Mobile Computing“ bewegen sich die Endgeräte physikalisch mit ihren Benutzern und mit ihnen der Zugang zu den zentralen IT-Diensten. Die angebotenen IT-Dienste unterscheiden sich beim Mobile Computing nicht von den stationär angebotenen Diensten.

Pervasive Computing

Im Unterschied zum Mobile Computing nutzen die Geräte des Pervasive Computing Informationen aus der und über die Umgebung, in der sie sich gerade befinden und reagieren darauf. Diese Geräte sind dadurch „kontextsensitiv“.

[6] pervasive: engl. überall vorhanden

[7] "Convenient access, through a new class of appliances, to relevant information with the ability to easyly take action on it when and where you need to."

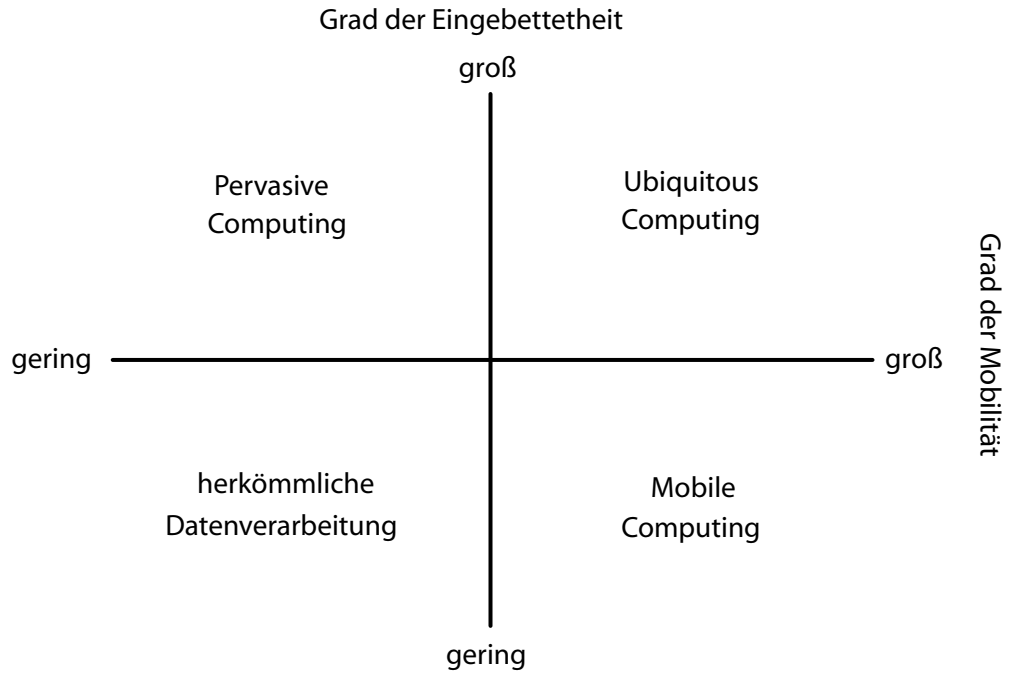

Abbildung 2.4.: Ausprägungen von Ubiquitous Computing, wobei die Einteilung nach dem Grad der Eingebettetheit und dem Grad der Mobilität vorgenommen wurde (nach Lyytinen und Yoo 2002, Seite 64).

Ubiquitous Computing

Beim Ubiquitous Computing sind die Geräte auch kontextsensitiv. Hier werden allerdings die technischen Fragen der Mobilität und der Eingebettetheit[8] einheitlich angegangen. So soll es bei Ubiquitous Computing unerheblich sein, ob sich die *Geräte* mit dem Nutzer bewegen, oder ob die *Anwendungen* dem Benutzer folgen, in dem sie sich von Gerät zu Gerät bewegen (Banavar und Bernstein 2002, Seite 93).

Im Folgenden werden „Mobile Computing“, „Pervasive Computing“ und „Ubiquitous Computing“ als feste Begrifflichkeiten weiterverwendet.

2.4. Existierende Lösungen

Um einen Überblick der verwandten Problemlösungen zu erhalten, sollen nun kurz ähnlich gelagerte Projekte vorgestellt werden.

[8] von engl. „Embeddedness“

infoAsset-Broker

Der infoAsset-Broker wurde für Organisationen entwickelt (infoAsset 2004). Das Wissen der Organisation soll mithilfe des infoAsset-Brokers zwischen den Mitarbeitern geteilt werden. Das Konzept folgt einem daten- und relationengetriebenen Ansatz, um strukturierte und unstrukturierte Informationen vernetzt abzuspeichern. Wissen wird durch Segmentierung und Klassifikation sprachlicher Einheiten, anhand von Taxonomien, kategorisiert.

MyLifeBits

Bush (1945) beschreibt einen „Apparat, in dem die Einzelperson alle persönlichen Bücher, Unterlagen und die persönliche Kommunikation aufbewahrt. Der Apparat ist mechanisiert, sodass sehr schnell und gezielt in diesem Archiv nachgeschlagen werden kann.“[9] Vannevar Bush nennt diese Einrichtung „Memex“.

Gemmell u. a. (2002) haben sich mit dem Forschungsprojekt „MyLifeBits“ der Vision von Bush (1945) angenommen (Microsoft 2004). Das MyLifeBits Projekt fokussiert ausschließlich das persönliche Wissen. Es soll alle Dinge, die ansonsten im Schuhkarton aufbewahrt werden, in digitaler Form speichern. Das beginnt bei Briefen und Fotos und hört bei Videofilmen nicht auf. Auch Telefonate sollen als Audiodatei gespeichert werden. Alle Daten werden als Binärobjekte (BLOB) in einer Datenbank abgelegt. Die interne Struktur der Daten wird außer Acht gelassen.

Haystack

„Haystack“ soll dabei helfen, die sprichwörtliche Nadel in einem Heuhaufen zu finden (Kargar u. a. 1999; Karger 2003). Das Programm vereint eine Vielzahl von unterschiedlichen Programmen unter einer Oberfläche. Es können persönliche Dokumente, wie Texte, Bilder, Adressen oder E-Mails verwaltet und durchsucht werden. Haystack benutzt zum Analysieren von unstrukturierten Texten Techniken aus dem Bereich der Künstlichen Intelligenz, um bessere Ergebnisse beim Wiederauffinden von Informationen zu erzielen. Die gespeicherten Informationen werden mittels des aus dem Semantic Web stammenden Resource Description Framework (RDF) beschrieben. Wie bei den im Abschnitt 2.2 beschriebenen Ontologien wird bei Haystack ein Modell erstellt. Dieses muss vom Anwender erstellt und gepflegt werden, was mit zusätzlichem Aufwand verbunden ist.

[9] "A memex is a device in which an individual stores all his books, records, and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility" (Bush 1945, S. 6).

Schlussfolgerung

Die existierenden Lösungen sind entweder auf die Bedürfnisse von Organisationen zugeschnitten oder sie vernachlässigen den Aspekt, dass Daten auf unterschiedlich gearteten Geräten verarbeitet werden müssen. Die hier vorgestellten Projekte MyLifeBits und Haystack sind zwar für Einzelpersonen konzipiert. Die Verteilung der Daten wird jedoch nicht behandelt. Die organisationsspezifischen Systeme sind für Einzelpersonen, aufgrund deren andersartigen Bedürfnisse, nicht praktikabel. Der infoAsset-Broker ist Vertreter dieser Kategorie.

Ein derartiges Hilfsmittel speziell für eigenständig agierende Wissensarbeiter soll in dieser Arbeit entworfen werden.

2.5. Zusammenfassung

Angesammeltes Wissen kann als Kapital betrachtet werden. Diesen Wert gilt es zu bewahren und zu nutzen. Einmal erworbenes Wissen sollte nicht dadurch verloren gehen, dass es nicht mehr auffindbar ist. Um das persönliche Wissen Gewinn bringend einsetzen zu können, muss es auch dort abrufbar sein, wo es benötigt wird: überall.

3. Analyse der Aufgabenstellung

Nachdem die Notwendigkeit des Wissensmanagements für Einzelkämpfer dargestellt wurde, sollen nun die Anforderungen an einen Lösungsansatz ermittelt und beschrieben werden. Begonnen wird mit der Betrachtung der Ausgangslage. Wie werden in einer Ich-AG Informationen abgelegt? Wie könnte die Ich-AG besser mit ihrer Ressource Information umgehen? Anhand von Szenarios wird dann die generelle Zielsetzung dargestellt und als Vision vom späteren Gesamtsystem näher beleuchtet.

Bevor die Anwendung modelliert werden kann, muss zunächst die Anwendungsdomäne bekannt sein. Dazu werden die Begrifflichkeiten aus dem Anwendungskontext betrachtet. Informationen sollen mit dem zu entwickelnden System verwaltet werden. Somit sind die Objekte, die die Informationen im System repräsentieren, auch die wichtigsten Objekte. Um die unterschiedlichen Aspekte von Informationsobjekten zu verdeutlichen, wird ein Analysemodell von ihnen erstellt.

Die wichtigsten Funktionalitäten des zu entwickelnden Systems werden mittels Anwendungsfällen beschrieben.

3.1. Ausgangspunkt (Ist-Zustand)

Im Büroalltag werden heutzutage für die unterschiedlichen Aufgaben viele verschiedene Programme, sogar verschiedene Geräte eingesetzt. Zur Durchführung einer Besprechung wird zuerst ein passender Termin mithilfe des Kalenders, einer so genannten Personal Information Manager (PIM) Anwendung, herausgesucht. Anschließend können die Telefonnummern oder E-Mail-Adressen der Teilnehmer aus dem Programm auf dem Arbeitsplatzrechner zusammengetragen werden. Vielleicht ist die eine oder andere Adresse nicht im Arbeitsplatzrechner verzeichnet oder nicht mehr aktuell. Dann könnte auch ein Handheld Computer benutzt werden, um Adressen zu suchen. Die Teilnehmer können nun telefonisch oder per E-Mail zur Besprechung eingeladen werden. Während der Besprechung werden Präsentationen vorgetragen. Die erarbeiteten Ergebnisse werden mithilfe von Textverarbeitung in einer Niederschrift zusammengefasst und an die Teilnehmer verschickt.

Bereits vor Jahren haben die Anbieter von Office Software erkannt, dass im Büro viele verschiedene Programme benötigt werden. Die Programme werden dann auch als Pakete zusammengefasst. So kann die Textverarbeitung, das Tabellenkalkulationsprogramm und das Präsentationsprogramm im Paket mit einem PIM erworben werden. Teilweise werden sogar alle Programme von einer gemeinsamen Oberfläche aus bedient. Die Arbeitsergebnisse werden jedoch nach wie vor nach Anwendungen getrennt gespeichert. Es ist nur sehr eingeschränkt möglich, von einem Dokument einer Anwendung auf Dokumente anderer Anwendungen zu verweisen.

Heute wird das digital vorliegende Wissen typischerweise auf unterschiedlichen Geräten, an unterschiedlichen Orten und in unterschiedlichen Formaten gespeichert. Durch die Verteilung auf mehrere Geräte und Formate ist eine Suche über das gesamte Wissen nur schwer möglich. Durch die redundante Datenhaltung treten auch leicht Inkonsistenzen der Daten auf den unterschiedlichen Geräten auf. Auch können zusammengehörende Daten nicht miteinander verknüpft werden.

Resümierend sollen die wesentlichen Probleme nachfolgend noch einmal aufgeführt werden:

- Speicherung von Informationen erfolgt momentan
  - auf unterschiedlichen Geräten
  - in unterschiedlichen Darstellungsformaten
  - an unterschiedlichen Orten
- Aufgrund der Verteilung der gespeicherten Informationen ist eine Verknüpfung der Daten nur schwer möglich.
- Eine Suche über das gesamte Wissen wird hierdurch erschwert.

3.2. Szenario (Soll-Zustand)

In hypothetischen Szenarios soll nun das Erscheinungsbild einer möglichen Lösung vermittelt werden.

Kai und Michael haben in Kais Büro am Berliner Tor eine Besprechung. Beide arbeiten mit UbiMex. Das bedeutet, auf Geräten des Ubiquitous Computing läuft diese Software. Die Geräte erkennen somit die Umgebung und beziehen aus ihr Informationen. Während Kai seinen Terminkalender sorgfältig führt und nicht nur Datum und Zeit der Besprechung einträgt, sondern auch Thema, Teilnehmer und Ort, steht bei Michael nur das Nötigste: Datum, Zeit und

„Projektbesprechung“. Die Eintragungen reichen zwar aus, um den Termin nicht zu verpassen, aber später kann anhand dieser spärlichen Daten nicht mehr rekonstruiert werden, was genau an dem Termin geschah. Der Terminkalender ist von immanenter Bedeutung, da er als Einstieg und Navigationshilfe für das persönliche Wissen dienen soll. Daher ist es unbedingt erforderlich, dass an dieser Stelle die Informationen sowohl quantitativ als auch qualitativ auf hohem Niveau erfasst werden. Das bedeutet, dass nicht nur viele verschiedene Daten eingegeben werden, sondern auch wertvolle Informationen von hoher Güte.

Da das System seine Umgebung erkennt und registriert, erschließt es sich die Kontextinformationen automatisch. Der Terminkalender erkennt seine Umgebung. Während des Termins wird der Ort erkannt und automatisch Berliner Tor dem Termin zugeordnet. Da Kai auch UbiMex benutzt, registrieren sich die beiden Geräte. Kais Gerät gibt sich als solches zu erkennen. Auf diese Weise wird Kai ohne weiteres Zutun dem Termin zugeordnet. Während der Besprechung werden verschiedene Dokumente geöffnet. Eine Powerpoint Präsentation und eine Excel Tabelle. Das registriert UbiMex und setzt diese mit dem Termin in Verbindung.

Die Vorteile der erstellten Verbindungen zeigen sich beim Wiederauffinden von Informationen.

Wiederauffindbarkeit

Die bereits erwähnte spezielle Funktion in einer umfangreichen Klassenbibliothek kann auf unterschiedliche Arten gefunden werden. Vielleicht kann der ungefähre Zeitraum eingegrenzt werden. Zusätzlich ist noch bekannt, in welchem Projekt die Klassenbibliothek eingesetzt wurde oder welche Personen im Zusammenhang mit der Klassenbibliothek stehen. Es muss nur ein Eintrag aus dem Kontext der gesuchten Information gefunden werden. Da Informationen, die in einem Sinnzusammenhang zueinander stehen, auch miteinander verknüpft sind, ist es nun möglich, sich zum gesuchten Funktionsnamen durchzuhangeln.

Wie kann die Telefonnummer einer Person ermittelt werden, von der nur noch der Vorname bekannt ist? Im Adressbuch kann nicht nachgeschaut werden. Dies ist nach Nachnamen sortiert. Doch es gibt zu viele Einträge mit dem gesuchten Vornamen. Glücklicherweise wird sich daran erinnert, wo die Person das letzte Mal getroffen wurde. Auch sind weitere Personen bekannt, die beim letzten Treffen mit anwesend waren. Da die Geräte beim Ubiquitous Computing Zugriff auf Informationen über die Umgebung haben, in der sie sich befinden, besteht die Möglichkeit diese auch als Metadaten abzuspeichern. So kann auch der Ort, an dem die Informationseinheiten benutzt wurden oder an dem ein Termin stattfand und die Personen, die auch anwesend waren, gespeichert werden. Schon sind genügend Suchkriterien vorhanden, um zum Ziel zu gelangen.

Bei der Suche nach einem bestimmten Bericht wird sich zum Beispiel nicht mehr an den Titel erinnert. Es wird sich jedoch noch daran erinnert, dass der Bericht in einer Besprechung vorgetragen wurde. Es sind möglicherweise noch einige der an der Besprechung beteiligten Personen bekannt. Diese Informationen ermöglichen es, die Besprechung im Terminkalender wieder zu finden. Wenn jetzt im Terminkalender beim entsprechenden Eintrag der Besprechung ein Verweis auf das dort erörterte Dokument existiert, ist der Bericht gefunden.

3.3. Vorteile

An dieser Stelle soll noch einmal rekapituliert werden. Wir befinden uns auf dem Wege in die sogenannte Wissensgesellschaft. Das bedeutet, dass Wissen immer wichtiger wird. Außerdem entwickelt sich der Einzelne immer mehr zum Einzelkämpfer, der völlig auf sich allein gestellt ist, und sich gegen alle anderen behaupten muss. UbiMex hilft durch die systematische und strukturierte Ablage von Wissen beim effizienten Arbeiten. Dabei bedeutet es keinen großen Mehraufwand. Bietet aber eine enorme Zeitersparnis, nebenbei spart es auch noch Nerven, da die aufwendige Suche entfällt. Die Effizienz ermöglicht es dem Wissensarbeiter, seine Kosten niedrig zu halten. Dieses wiederum sichert die Wettbewerbsfähigkeit: Arbeitszeit wird nicht mit der Suche nach Informationen vertan, sondern kann direkt in das Projekt fließen. Dieses System gewährleistet, dass der Ich-AGler jederzeit in der Lage ist, dem Kunden kompetente Hilfe anzubieten und sich schnell auf jede Situation einstellen kann. Dieses führt zu einer engen Kundenbindung und sichert letztendlich die Existenz des Wissensarbeiters.

3.4. Anforderungen

Der Weg vom Ist-Zustand zum Soll-Zustand, wurde nun beschrieben. Im folgenden Abschnitt werden die Anforderungen an ein System erarbeitet.

Das persönliche Wissen muss dort, wo es benötigt wird, auch zur Verfügung stehen. Um eine Allgegenwärtigkeit des persönlichen Wissens herzustellen, muss das auf unterschiedliche Geräte verteilte Wissen zuerst zusammengefasst werden. Das zusammengefasste Wissen muss nun auf den unterschiedlichen Geräten auch angezeigt werden können. Da die Informationen aus den unterschiedlichsten Quellen stammen, liegen die Daten auch in verschiedenen Datenformaten vor. Somit müssen unterschiedliche Datenformate im- und exportiert werden können. Das persönliche Wissen muss dann auch so strukturiert werden, dass es vom Eigentümer auch schnell wieder gefunden wird. Die Punkte Wissensstrukturierung, Datenabgleich und Transformation werden in den folgenden Abschnitten noch einmal etwas

ausführlicher behandelt. Zunächst sollen aber noch weitere allgemeine Anforderungen betrachtet werden.

Da die Einzelkämpfer, für die das System entwickelt werden soll, typischerweise nicht über das Budget einer Organisation verfügen, müssen die Kosten möglichst gering gehalten werden. Trotzdem muss jedoch eine zuverlässige Aufbewahrung des persönlichen Wissens unbedingt gegeben sein. Denn die Informationsobjekte stellen das Kapital dar und somit ist die Informationssammlung entsprechend wertvoll. Die Informationsobjekte sollten dem Wert und ihrem Inhalt entsprechend sicher aufbewahrt werden.

Der Bereich Datensicherheit unterteilt sich in zwei Aspekte. Zum einen in den Schutz der Daten vor unberechtigtem Zugriff. Zum anderen müssen die Daten vor Verlust geschützt werden. Es dürfen keine Verfälschungen der Daten aufgrund technischer Fehler möglich sein. Weder ein Spannungsausfall noch ein Plattenfehler darf die Daten gefährden. Im Fehlerfall muss eine Wiederherstellung immer möglich sein.

Wissensstrukturierung

Informationen müssen derartig abgelegt werden, dass sie leicht wiederauffindbar sind. Die Wiederauffindbarkeit darf allerdings keinen zu großen Mehraufwand bedeuten. Auch muss neues Wissen ohne großen Aufwand schnell und unkompliziert hinzugefügt werden können. Der Aufwand muss so gering sein, dass auch nach einer langen Suche nach der passenden Funktion in der Klassenbibliothek, die Motivation weiterhin vorhanden ist, die gefundene Erkenntnis zu archivieren.

Transformation

Je nach Einsatzzweck können unterschiedliche Geräte zum Einsatz kommen. Jedes dieser Geräte erwartet meist ein eigenes, auf das Gerät abgestimmtes, Darstellungsformat. Auch durch die Verwendung von unterschiedlichen Programmen auf demselben Gerät liegen Informationen in unterschiedlichen Formaten vor. Die Daten sollten in einem einheitlichen Format abgespeichert werden. So kann das Ausgabeformat dynamisch, je nach Bedarf, aus den gleichen Daten erzeugt werden. Zur einfacheren Handhabung der unterschiedlichen Ausgabeformate wird eine Trennung von Struktur, Inhalt und Präsentation der Daten angestrebt.

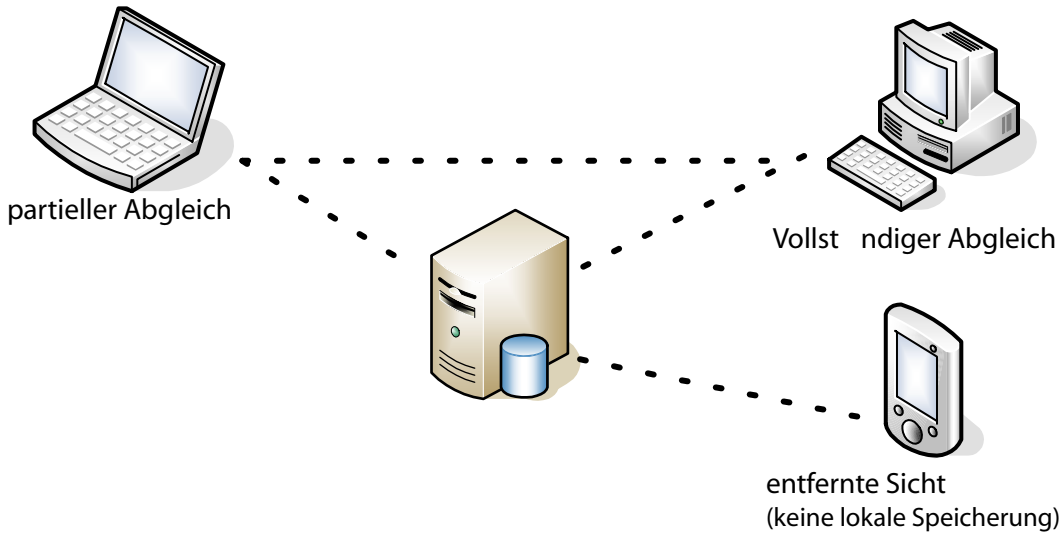

Abbildung 3.1.: Datenabgleich

Datenabgleich

Was nützt das größte Archiv, wenn die Informationen nicht dort sind, wo sie benötigt werden? Der Zugriff auf die archivierten Informationen muss somit überall möglich sein. Wirklich allgegenwärtige Verfügbarkeit der Informationen kann nur mittels mobiler Endgeräte erreicht werden. Je nach Einsatzzweck werden unterschiedliche Geräte für den Zugriff auf die Informationen verwendet. Von unterschiedlichen stationären Arbeitsplatz PCs oder portablen Notebooks, genauso wie von mobilen Handheld Geräten. Vor allem mobile Geräte werden jedoch – schon aus Platzgründen – immer weniger Speicherplatz zur Verfügung haben, als stationäre Systeme. Somit muss berücksichtigt werden, dass gegebenenfalls nur Teildatenbestände mitgenommen werden können. Daher besteht beim partiellen Datenabgleich zusätzlich die Problematik, nur die aktuell relevanten Informationen auf das Gerät zu laden.

Selbst wenn eine noch so gute Methode zur Auswahl der relevanten Informationen verwendet wird, ist nach Murphys Gesetz die gerade wichtigste Information nicht übertragen worden. Falls unterwegs Daten benötigt werden, die sich nicht auf dem mobilen System befinden, sollten diese per Online-Anfrage vom stationären System auf das mobile Gerät übertragen werden. Für Fernabfragen sind die Datenraten, der schon heute am Markt befindlichen Datenprotokolle auf Basis der GSM-Mobilfunktechnik (GPRS, HSCSD und EDGE), ausreichend hoch (Beyer u. a. 2001). Und auch für die Kommunikation im Nahbereich sind die nötigen Techniken verfügbar (Bluetooth und WLAN).

Als Alternative zum vollständigen oder auch nur partiellen Abgleich der Daten können zusätzlich entfernte Sichten auf zentral gespeicherte Daten verwendet werden. Die auf einem

entfernten Gerät gespeicherten Daten werden nur auf dem lokalen Gerät angezeigt, nicht jedoch auf das lokale Gerät geladen.

Somit werden die folgenden Datenabgleichsarten benötigt:

- Vollständiger Abgleich
- Partieller Abgleich
- entfernte Sicht auf die Informationen

Zu unterscheiden ist noch, ob die Daten auf allen Geräten auch verändert werden können sollen. Wenn es möglich sein soll, Daten unterwegs zu ändern und die geänderten Daten wieder den anderen Geräten zur Verfügung zu geben, dann muss ein nicht unerheblicher Aufwand getrieben werden, um die unterschiedlichen Datenbestände synchron zu halten. Dieser Aspekt soll hier nicht weiter vertieft werden, da eine erschöpfende Betrachtung des Themas über den Rahmen dieser Arbeit hinausgehen würde.

3.5. Nichtfachliche Anforderungen

Neben den fachlich motivierten Anforderungen haben sich in der Informatik allgemein anerkannte Prinzipien herausgebildet, die eine gute Softwarearchitektur auszeichnen. Die Beachtung dieser Grundsätze soll dazu führen, dass die Architektur zuverlässig, wartbar, anpassbar und effizient ist. Obwohl ein Nachweis über die Brauchbarkeit dieser Prinzipien fehlt, gelten sie trotzdem als allgemein anerkannt. Die Akzeptanz dieser Grundsätze beruht einzig auf (überwiegend) positiven Erfahrungen.

Einen der ersten Ansätze, die Entwurfsprinzipien zusammenzufassen, haben Ross u. a. (1975) gemacht. Die von Ross u. a. aufgeführten Prinzipien finden sich auch in späteren Aufzählungen - teils abgewandelt - immer wieder. Zum Beispiel geht Buschmann u. a. (1996, Seite 397) ausführlich darauf ein.

Ein zuverlässiges System kann nur erstellt werden, wenn die Komplexität des Systems beherrschbar ist. Um auch umfangreiche Systeme erfassbar zu halten, kann ein komplexes Gesamtsystem in kleine, möglichst unabhängige, Teilsysteme aufgegliedert werden. Auf diese Art müssen immer nur Probleme von relativ geringer Komplexität auf einmal gelöst werden. Die weniger komplexen Teilprobleme bleiben handhabbar. Aus den einzelnen Teillösungen wird dann die Gesamtlösung zusammengesetzt. Wobei der innere Aufbau der einzelnen Teile abstrahiert werden kann. Wirth (1971) nennt diesen Ansatz „schrittweise Verfeinerung“.

Ein derartiger iterativer und evolutionärer Ansatz hat neben der Beherrschung der Komplexität auch taktische Vorteile. Die Annahme, dass der Anwendungsbereich zu einem bestimmten Zeitpunkt komplett verstanden ist, ist in der Praxis nur selten erfüllt. Es kommt auch eher selten vor, dass ab einem gewissen Zeitpunkt keine Änderungen der Anforderungen mehr eintreten (Lorenz 1993, Seite 8f). Daher erscheint es auch nicht gerade sinnvoll, im ersten Schritt eine zu erstellende Anwendung komplett zu modellieren. Anfänglich werden vielleicht die Schwerpunkte falsch gesetzt. So würde viel Energie in die Realisierung von Bereichen verwendet, die sich später als nicht erforderlich herausstellen. Ein sich allmählich entwickelndes System hat zusätzlich den Vorteil, dass bereits mit frühen Versionen gearbeitet werden kann. Häufig wird auch erst durch die konkrete Arbeit mit dem System die Arbeitsumgebung richtig erfassbar.

Somit sollten zuerst die Kernfunktionalitäten des Systems identifiziert werden. Im ersten Schritt wird nur diese umgesetzt. Zusätzlich werden sämtliche Hilfsfunktionen implementiert, die nötig sind, um das System bereits zu nutzen. So kann das System langsam wachsen. Die einfachste Lösung ist daher einer Komplizierteren vorzuziehen. Ist die einfache Lösung erst einmal umgesetzt, so kann sie immer noch erweitert werden. „Wenn Einfachheit gut ist, dann wählen wir stets das System, welches am einfachsten aufgebaut ist und die geforderte Funktionalität unterstützt“[1] (Beck 1999).

Zusätzlich zum schrittweisen Hinzufügen der Funktionalität kann die Entwicklung, Änderung und Wartung des Systems auch durch eine gute Zerlegung in Module erleichtert werden.

Laut Parnas (1972) sollen Entwurfsentscheidungen vor dem Rest des Systems verborgen werden. Das Zurückhalten von Informationen hilft dabei, die Kopplung der Module untereinander zu reduzieren (Simon 1962). Durch das Verbergen der Entwurfsentscheidungen steht weniger Wissen zur Verfügung, das zu unnötigen Abhängigkeiten untereinander hätte führen können. Die Entkopplung der unterschiedlichen Bereiche verhindert die expotenzielle Zunahme von Abhängigkeiten zwischen den Klassen, wenn weitere Komponenten hinzugefügt werden. Änderungen wirken sich ebenfalls nur lokal aus.

Neben den oben aufgezählten Grundsätzen sollte, wann immer es möglich ist, „das Rad nicht neu erfunden werden“. Vielmehr sollte für eine vorgesehene Komponente eine bereits vorhandene Komponente verwendet werden. Neben der Kosten- und Zeitersparnis hat Wiederverwendung den Vorteil, dass die Komponente bereits erprobt ist und somit die Fehlerwahrscheinlichkeit niedriger ist, als bei einer neu erstellten Komponente. Die Neuerstellung hat lediglich den Vorteil, dass die Komponente völlig nach den eigenen Vorstellungen erstellt werden kann, während bei der Wiederverwendung unter Umständen Kompromisse nötig sind.

[1] "If simplicity is good, we’ll always leave the system with the simplest design that supports its current functionality."

3.6. Betrachtung der Verfahren zur strukturierten Ablage von Wissen

Im Grundlagenkapitel wurden im Abschnitt 2.2 unterschiedliche Möglichkeiten zur strukturierten Ablage von Wissen vorgestellt. Die vorgestellten Verfahren sollen nun insbesondere hinsichtlich ihrer Eignung in Bezug auf das Management von persönlichem Wissen betrachtet werden, um anschließend ein Verfahren auszuwählen.

Klassifikationssysteme und Ontologien

Im Hinblick auf Ich-AGs haben Klassifikationssysteme, neben dem hohen Aufwand sie zu erstellen und sich in sie einzuarbeiten, einen entscheidenden Nachteil: Am Beginn des Berufslebens ist ein Überblick über das gesamte spätere Themengebiet kaum möglich. Es kann somit unter Umständen kein Klassifikationssystem erstellt werden. Ein weiteres Problem in Bezug auf Kategorisierung besteht in der nicht eindeutigen Klassifizierbarkeit von Informationen. Die spezielle Funktion in einer umfangreichen Klassenbibliothek, nach der in einem vorangegangenen Projekt bereits lange gesucht wurde, kann sowohl unter der Einordnung „verwendete Programmiersprache“, als auch unter der Kategorie des Projekts gesucht werden. Ähnliches, wie bei Klassifikationssystemen, gilt auch für Ontologien. Hier müsste zuerst das abstrakte Modell entworfen werden.

Semantische Verweise

Aus den oben genannten Gründen wird sich gegen die Verwendung von Klassifikationssystemen und Ontologien ausgesprochen. Stattdessen wird zur Ablage von persönlichem Wissen ein Verfahren benötigt, das ohne umfangreiche Vorarbeiten auskommt. Auch sollte die Tatsache berücksichtigt werden, dass abzuspeichernde Informationen nicht immer eindeutig klassifizierbar sind. Informationen bestehen aus Fakten, die mit dem Kontext verknüpft sind, in dem sie stehen. Wenn dies mithilfe des Ablagesystems abgebildet werden kann, würde sich eine Struktur innerhalb der Informationen quasi von selbst ergeben. Semantische Verweise hätten den Vorteil, dass das System „wissen“ würde, welche Informationen in welcher Beziehung zueinander stehen. Das System könnte dem Anwender intelligentere Suchhilfen anbieten. Bei semantischen Verweisen besteht jedoch das schon im Zusammenhang von Klassifikationssystemen beschriebene Problem: Die Domäne muss zunächst modelliert werden. Dies ist für ein persönliches Wissensmanagementsystem nur schwer möglich.

Assoziative Verknüpfungen

Wie oben beschrieben, können die herkömmlichen – in Organisationen eingesetzten – Verfahren hier nicht verwendet werden. Denn ein alleine agierendes Individuum hat andere Anforderungen als eine Organisation. Die von Organisationen verwendeten Verfahren fordern dem Nutzer einen gewissen Aufwand ab, der von einer Organisation ohne Probleme zu bewältigen ist, aufgrund der höheren Ressourcen. Es wird schlichtweg eine Archivabteilung gegründet. Dieser Aufwand ist jedoch von einer Einzelperson nicht zu leisten. Die Einzelperson hat die Vorteile dieses erhöhten Aufwandes zudem nicht, denn die dadurch entstehenden Vorteile zielen vor allem auf Wissenstransfer ab.

Ein weiterer Aspekt besteht in der Problematik, dass in Organisationen jeder einzelne Mitarbeiter sein eigenes System des Wissensmanagements hat. Dieses Wissen soll eben nicht nach dem individuellen System abgespeichert werden, sondern in einem standardisierten, organisationseigenen Verfahren. Der Aufwand besteht nun in der Nivellierung der Teilsysteme. Beim persönlichen Wissensmanagement gibt es keine Teilsysteme. Von daher fällt der Aspekt der Nivellierung weg. Ebenso entfällt der Wissenstransfer. Der Transfer von Wissen findet höchstens in der Weise statt, dass das Wissen als Tauschobjekt zwischen Partnern ausgetauscht wird. Es handelt sich hierbei nicht um einen automatisierbaren Vorgang. Es muss lediglich die Möglichkeit vorgesehen werden, Datenbestände manuell zu im- und exportieren.

Als Alternative zu den gängigen Methoden der strukturierten Ablage von Wissen könnte die Natur zum Vorbild genommen werden. In der psychologischen Fachwelt wird angenommen, dass Informationen, die einmal im Gedächtnis abgespeichert worden sind, dort für immer gespeichert bleiben (vgl. z. B. Linton 1975). Wenn sich nicht mehr an bestimmte Informationen erinnert werden kann, ist dieser Theorie zufolge lediglich der Weg zur Information verloren gegangen. Experimente legen den Schluss nahe, dass Wissen im Langzeitgedächtnis hierarchisch organisiert gespeichert wird. Dies erfolgt in so genannten Semantischen Netzen. Diese Bedeutungsnetzwerke ermöglichen es uns, gespeicherte Informationen ausfindig zu machen (z. B. Collins und Quillian 1969). Die Informationen im Langzeitgedächtnis werden zusammen mit Informationen über den Kontext, in dem sie gelernt wurden, gespeichert (Schacter u. a. 1984).

Der Kontext ist als ein inhaltlicher Sach- und Situationszusammenhang zu verstehen, in dem die Information steht und aus der sie heraus verstanden werden muss. Diese zusätzlichen Informationen können beim Wiederauffinden von Informationen nützlich sein.

Informationen, die in einer Beziehung zueinander stehen, sollten miteinander verknüpft werden können. Zusätzlich sollte es möglich sein, weitere Informationen hinzuzufügen. Dokumente sollten auch nicht nur auf andere Dokumente verweisen können, sondern auch zum Beispiel Personen oder Orte. So soll es möglich werden, das Wissen zusammen mit dem

Sinnzusammenhang abzuspeichern. Der Mehraufwand zum Verwalten der archivierten Informationen muss minimal sein. Bei der späteren Suche nach einer bestimmten Information würde es somit ausreichen, in die inhaltliche Nähe der gesuchten Information zu gelangen. Von dort kann über die Verknüpfungen schnell zur gesuchten Information navigiert werden.

Auswahl eines Verfahrens

Der im Abschnitt 2.2 beschriebene erweiterte Hypertext erfüllt die oben gemachten Anforderungen. Informationen werden assoziativ miteinander verknüpft. Da lediglich syntaktische Verweise benutzt werden, muss im Gegensatz zu semantischen Verweisen kein Modell erstellt werden. Das Verfahren kann sofort eingesetzt werden. Informationen können somit zusammen mit ihrem Sinnzusammenhang abgepeichert werden, ohne dass hierfür Vorarbeiten, wie die Erstellung von Kategorien oder Modellen, nötig sind. Dem System können die Verknüpfungen nicht näher spezifiziert werden. Anhänge bieten die Möglichkeit, zusätzliche Informationen zu den vorhandenen Inhalten zu liefern. Dadurch könnte das System um intelligentere Verfahren erweitert werden. Ein weiterer Vorteil ist, dass Verweise in beide Richtungen benutzbar sind.

3.7. Anwendungsfälle

Die einzelnen Arbeitsabläufe, die für die softwaretechnische Umsetzung relevant sind, werden nun in Form von Anwendungsfällen dargestellt. Ein Anwendungsfall besteht aus einer Reihe von Abläufen, die vom Akteur ausgelöst werden. Zusätzlich dienen die Anwendungsfälle zur Abgrenzung des Systems.

Die Entwicklung soll, wie in Abschnitt 3.5 beschrieben, einen inkrementellen Ansatz verfolgen. In diesem ersten Schritt werden nur die Kernfunktionen realisiert. In weiteren Iterationsschritten kann dann nach und nach das System an Funktionalität wachsen (vgl. Beck 1999).

Anwendungsfall: „Einfügen“

Informationsobjekte in das System aufnehmen. Alle zusätzlich verfügbaren Daten über das einzufügende Informationsobjekt werden vom System automatisch hinzugefügt. Die Informationsobjekte können unterschiedlich ausgeprägt sein (Kontakt, Termin, Dokument . . .).

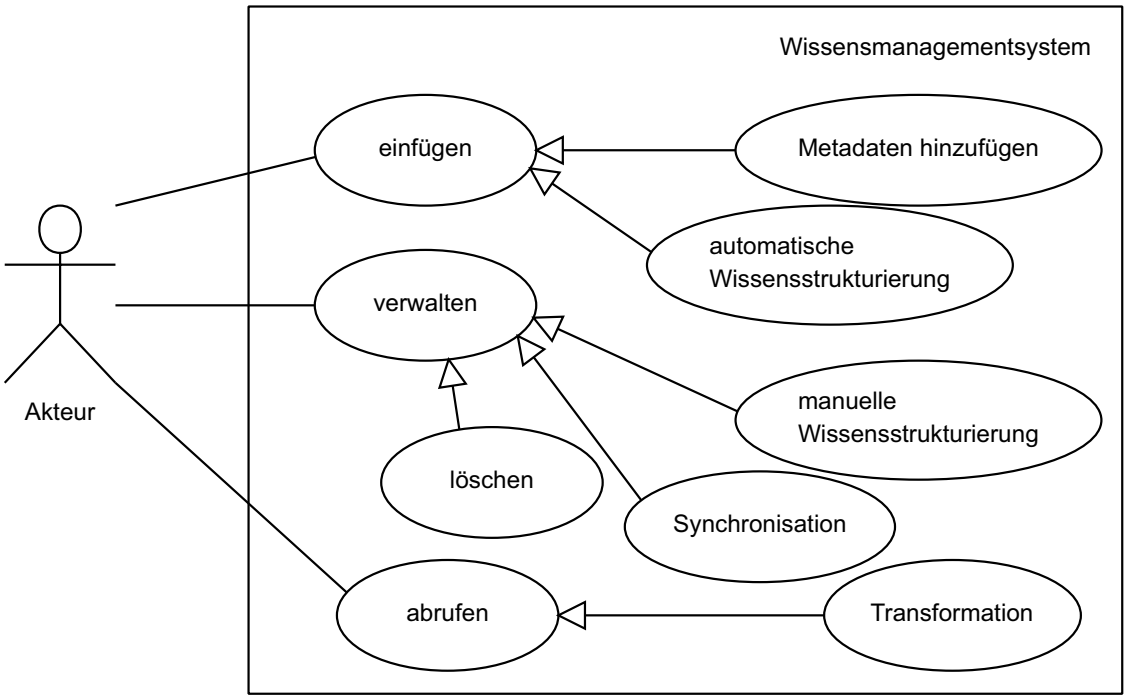

Abbildung 3.2.: Anwendungsfälle in UML-Notation

Anwendungsfall: „Verwalten“

Die im System befindlichen Informationsobjekte verwalten. Verknüpfungen zwischen Informationsobjekten hinzufügen. Die Datenbestände auf unterschiedlichen Geräten werden vom System synchron gehalten.

Durch das spätere Löschen von Informationsobjekten könnten „Wissensinseln“ entstehen. Wenn das gelöschte Informationsobjekt einen Teilbereich der gespeicherten Informationen mit den restlichen Informationen verknüpfte, so ist dieser Teilbereich nun schwerer oder nicht mehr erreichbar. Es würde genau das passieren, was im menschlichen Gehirn als „Vergessen“, oder korrekter als „Verdrängen“ bezeichnet wird. Die Information ist noch vorhanden, nur der Weg dorthin existiert nicht mehr. Es werden Hilfsmittel benötigt, um Ähnlichkeiten zwischen Dokumenten der Wissensinsel und dem Rest zu erkennen. Neue Verknüpfungen müssten dann eine Brücke zu den Wissensinseln schlagen. Dieser Aufwand soll – zumindest im ersten Schritt – nicht getrieben werden. Somit wird das Löschen nicht unterstützt. Was ist jedoch mit versehentlich eingefügten Daten? Eine Möglichkeit wäre es, nur Informationsobjekte löschen zu lassen, wenn die Information mit keiner anderen in Beziehung steht.

Anwendungsfall: „Abrufen“

Informationsobjekte aus dem System abrufen. Gegebenenfalls wird das Darstellungsformat auf das Gerät automatisch angepasst.

3.8. Zusammenfassung

Ein persönliches Wissensmanagementsystem soll Daten beliebiger Formate verarbeiten können. Die Informationen müssen ohne großen Aufwand in das System aufgenommen werden können, wobei bei der Ablage von neuen Informationen diese so abgelegt werden müssen, dass sie später von derselben Person auch wieder gefunden werden. Es wird die Möglichkeit der Replikation der Datenbestände zwischen unterschiedlichen Geräten benötigt. Dieser Aspekt soll hier allerdings nur sehr kurz behandelt werden, da eine umfassende Behandlung zu umfangreich für diese Arbeit wäre. Der Zugriff auf die Informationen muss von den unterschiedlichsten Geräten möglich sein. Das durch das System verwaltete persönliche Wissen muss zuverlässig aufbewahrt werden. Auch sollte die Lösung sich nicht auf einzelne Anbieter stützen. Vielmehr sollten offene Standards verwendet werden. Das Verfahren zur Ablage des Wissens soll so einfach wie möglich gehalten sein. Zusätzlich muss das System für eine Einzelperson bezahlbar sein.

Diese Arbeit stellt eine erste Näherung an ein persönliches Wissensmanagementsystem dar. Daher werden nur die Grundfunktionalitäten Einfügen, Verwalten und Abrufen betrachtet.

4. Entwurf einer möglichen Lösung

Das Ziel wurde erörtert. Nun wird eine Lösung erarbeitet. Zuerst wird der wichtigste Bestandteil des Systems, das Informationsobjekt, mithilfe eines Objektmodells beschrieben. Das zu realisierende Problem soll auf diese Art und Weise verständlicher gemacht werden. Dieses Modell beschreibt die wesentliche Struktur und Semantik des Problems. Nach der Detailbetrachtung wird die gesamte Anwendungslogik untersucht und anschließend die einzelnen Bestandteile der Anwendungslogik. Die dann in eine Systemarchitektur eingebettet wird.

4.1. Informationsobjekte

Die kleinste Einheit, in die eine Information hier aufgeteilt werden soll, wird im Folgenden als Informationsobjekt bezeichnet. Um eine Art der Ablage zu modellieren, müssen zunächst die Kennzeichen der Informationen betrachtet werden, um sie anschließend abzubilden. Informationen können nie im kontextfreien Raum stehen. Denn eine Einzelinformation steht niemals für sich alleine, sie wird immer begleitet von weiteren Faktoren, wie dem Ort, der Zeit, einer spezifischen Lebensphase oder Ähnlichem. Daher referenzieren Informationen immer auf weitere Informationen. Die Repräsentation der Informationsobjekte im System muss diesen Aspekt abbilden.

In Abbildung 4.1 ist ein Informationsobjekt modelliert. Ein Informationsobjekt besteht aus der jeweiligen Information, und den so genannten Metadaten, Informationen über diese Information. Erstellungsdatum oder Kontext der Information sind zum Beispiel solche Metadaten. Zu den Metadaten gehören auch persönliche Bewertungen und Anmerkungen. Die Informationen besitzen unterschiedliche Aspekte, aus denen sie heraus betrachtet werden können (Aufgaben, Termine). Neben den Informationsobjekten, die konkrete Aspekte abbilden, gibt es noch eine abstrakte Kategorie. Diese Gattung ist zum Beispiel für Dokumente vorgesehen. Diese können dann beispielsweise als Texte eine Struktur aufweisen. Grafiken und sonstige Binärobjekte werden als unstrukturierte Daten abgelegt. Eine einzelne Information kann aus einem Informationsobjekt bestehen. Sie kann aber auch aus mehreren zusammengesetzt sein. So ist ein Termin zum Beispiel eine Komposition aus Ort, Zeit und Person. Diese Komposition kann mittels der Assoziation „InfoObject – Composite“ abgebildet werden. Informationen besitzen die folgenden Bezugsarten:

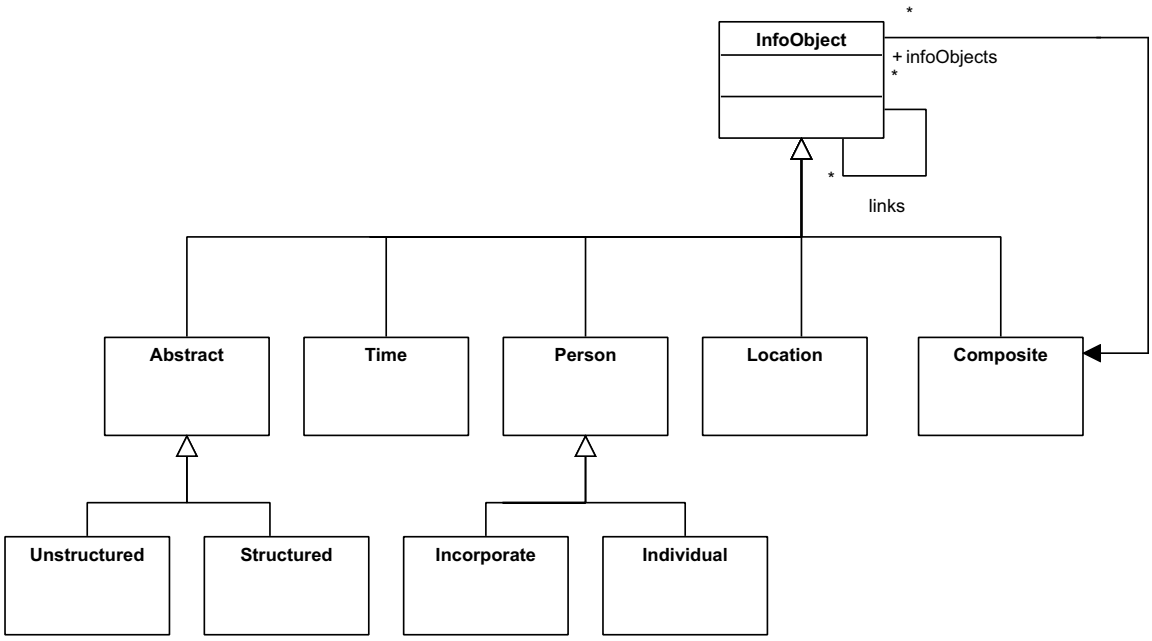

Abbildung 4.1.: UML-Analysemodell des Informationsobjekts

- Zeitbezug
- Ortsbezug
- Personenbezug
- Themenbezug

Einzelne Informationsobjekte können teilweise noch weiter unterteilt werden. Für eine Privatperson müssen beispielsweise geringfügig andere Informationen gespeichert werden, als für eine juristische Person.

4.2. Ein persönliches Wissensmanagementsystem für Ubiquitous Computing

Das im Abschnitt 3.6 zur Ablage der Daten gewählte Verfahren des erweiterten Hypertextes zur assoziativen Verknüpfung ist sehr einfach gehalten. Es schließt jedoch aufwendigere Verfahren nicht aus. Wie wird es konkret umgesetzt? Zunächst wird eine geeignete Datenstruktur entworfen. Diese Datenstruktur muss offen für unterschiedliche Verfahren sein. Danach folgt der Entwurf der Anwendungslogik mit seinen Hauptbestandteilen.

Vom Terminkalender zum persönlichen Informationsmanager

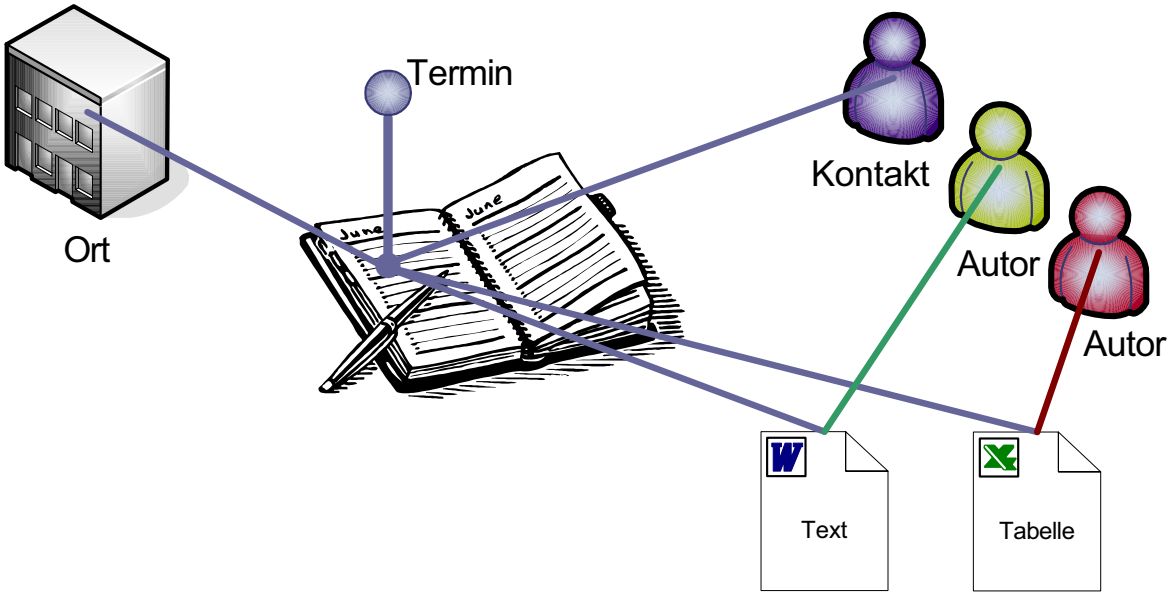

Abbildung 4.2.: Vom Terminkalender zum persönlichen Informationsmanager

Vom Grundprinzip her verknüpft der Terminkalender verschiedene Arten von Informationsobjekten. Diese Verknüpfungen erfolgen ohne großen Mehraufwand – quasi automatisch, da sie vom System vorgenommen wird. Es bedarf nur noch der Möglichkeit, die an diesem Termin behandelten Informationen mit dem Termin zu verbinden. Der Terminkalenderansatz muss nur minimal erweitert werden, um daraus einen universellen Informationsmanager zu machen.

Der Terminkalender verknüpft von sich aus bereits verschiedene Informationsobjekte. Abbildung 4.2 soll dies an einem Beispiel verdeutlichen. Hier verbindet ein Termin einen (weiteren) Kontakt mit einem Ort. Zwei Dokumente sind Thema des Treffens. Diese Dokumente werden zusätzlich mit dem Termin verbunden. An den verknüpften Objekten „hängen“ unter Umständen weitere Objekte. In Abbildung 4.2 sind die Dokumente nur mit ihren jeweiligen Autoren verknüpft. Zur vereinfachten Darstellung wurde darauf verzichtet, die komplexen Verknüpfungsnetze abzubilden.

Mithilfe der Verknüpfungen im erweiterten Kalender kann dieser als Einstiegspunkt für eine Suche benutzt werden. Als zusätzliche Möglichkeit der Suche im Datenbestand wird optional eine N-dimensionale Suchmaske angeboten. Hiermit können dann Suchanfragen gestellt werden, wie sie im Abschnitt 3.2 beschrieben sind.

Anwendungslogik

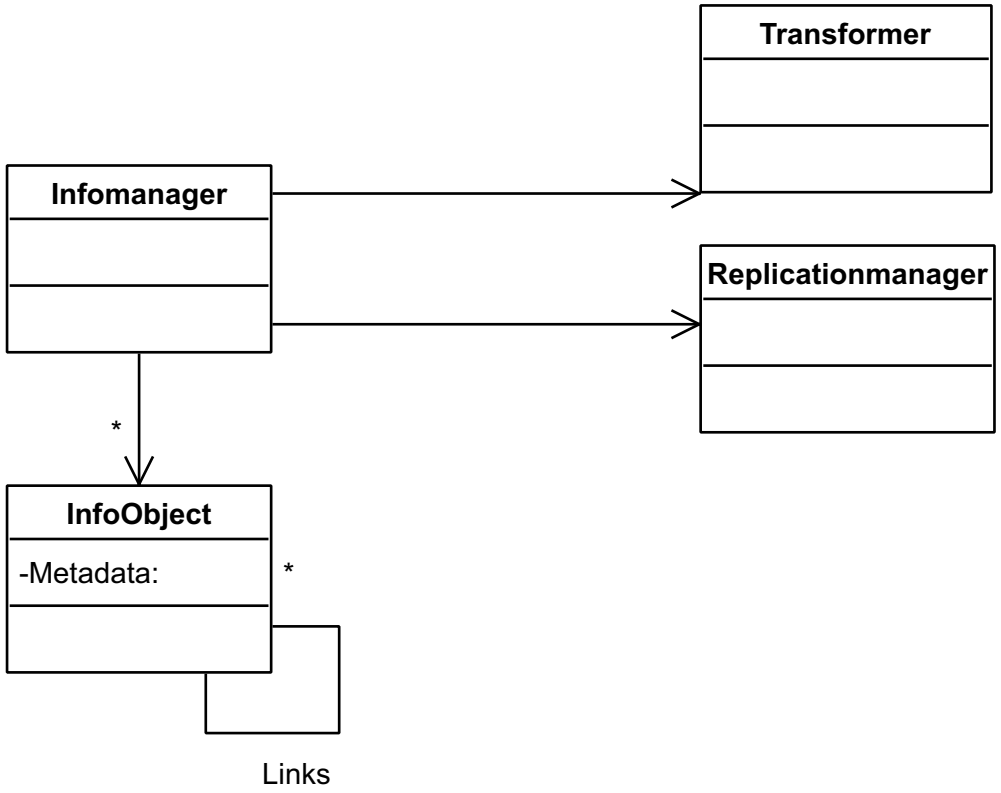

Abbildung 4.3.: UML-Klassendiagramm

Aus den Anforderungen ergeben sich drei Aufgabenbereiche. So müssen die Informationsobjekte verwaltet werden. Das übernimmt der erweiterte Kalender. Dieser muss zusätzlich noch die Informationen allgegenwärtig machen, wie im Abschnitt 3.4 gefordert. Somit müssen die Informationen in unterschiedliche Formate gewandelt werden können. Schließlich müssen unterschiedliche Datenbestände untereinander abgeglichen werden können. Daraus folgen die in Abbildung 4.3 dargestellten Bestandteile der Anwendungslogik `Infomanager`, `InfoObject`, `Transformer` und `Replicationmanager`.

Infomanager

Der `Infomanager` verwaltet die Informationsobjekte. Er verknüpft, soweit es möglich ist, automatisch Informationsobjekte, die in einer Beziehung zueinander stehen. Über ihn kann

der Anwender zusätzliche Verknüpfungen manuell erstellen und erläuternde Vermerke an Informationsobjekten anbringen. Da das persönliche Wissensmanagementsystem, wie bereits beschrieben, als erweiterter Terminkalender konzipiert sein soll, müssen einige Informationsobjekte vom System bearbeitet werden. Dokumente werden nicht näher vom System betrachtet. Die für einen Terminkalender typischen Objekte, wie Termine, Kontakte, Notizen oder Aufgaben, jedoch schon.

InfoObject

Die komplexe Abbildung der Wirklichkeit, die im Rahmen der Analyse in Abschnitt 4.1 dargestellt wurde, wird stark abstrahiert. Eigentlich wird nur ein Container für unterschiedlich geartete Objekte benötigt. Die Operationen auf den Objekten sind während der Verwaltung gleich. So muss es möglich sein, dass Informationsobjekte auf andere Informationsobjekte verweisen. Zusätzlich müssen Informationsobjekte Metainformationen über sich selbst speichern. So zum Beispiel ein Zeitstempel der letzten Änderung. Oder ein Zähler, der mitzählt, wie häufig das Informationsobjekt bearbeitet wurde.

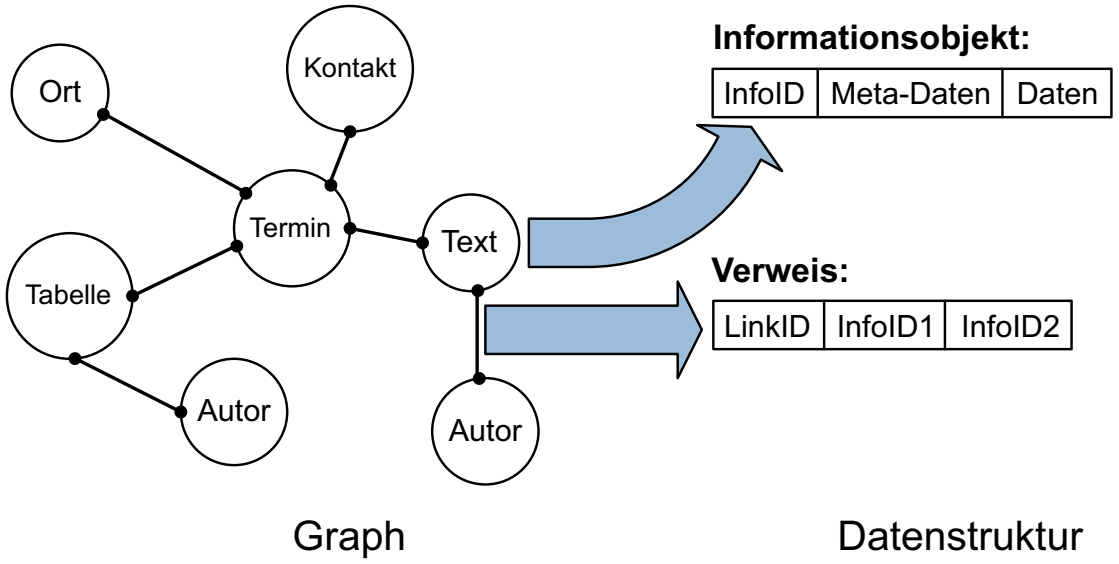

Abbildung 4.4.: Der Graph der Informationsobjekte und die entsprechende Datenstruktur

Derartig abgespeicherte Informationsobjekte bilden einen allgemeinen, ungerichteten Graphen, wie in Abbildung 4.4 dargestellt. Zur Ablage der Daten ist der Aufbau der Daten unerheblich. Es wird nur zwischen Daten und Metadaten unterschieden. Zusätzlich ist es erforderlich, dass die Verweise der Daten untereinander in der Datenstruktur abgebildet werden. Solange die Metainformationen nicht ausgewertet werden, sind die Kanten des Graphen ungefärbt. Mithilfe der Metadaten können die Kanten jedoch gewichtet werden. Eine mögliche

Gewichtung ist beispielsweise über die Häufigkeit der Nutzung möglich. Kanten zu häufig benutzten – und damit potenziell wichtigen – Informationsobjekten erhielten dann eine hohe Gewichtung.

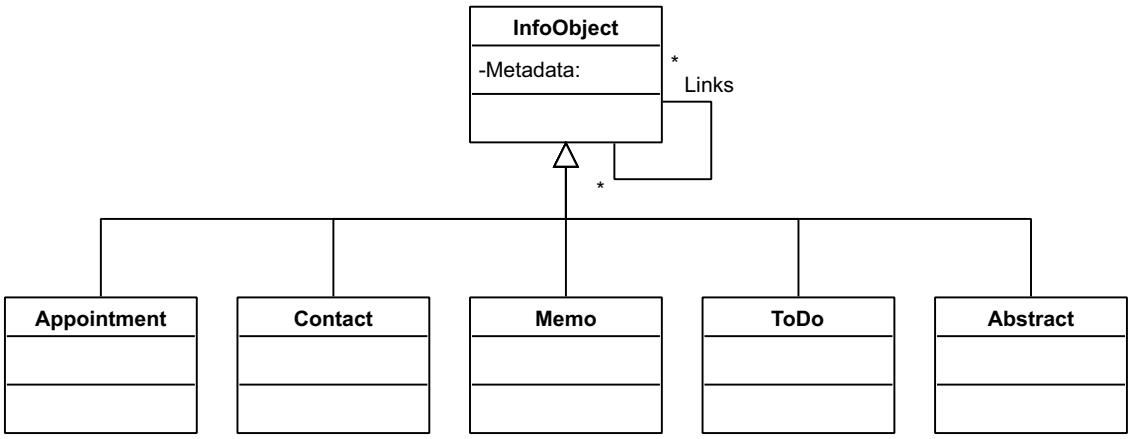

Abbildung 4.5.: UML-Klassendiagramm des vereinfachten Informationsobjekts

Neben den für einen Terminkalender typischen Objekten, wie Termine, Kontakte, Notizen oder Aufgaben, existiert noch ein Objekttyp für generische Objekte (vgl. Abb. 4.5). Die Basisobjekte werden vom System bearbeitet und werden als eigene Klassen modelliert. Die generischen Objekte sind für Informationen, die beispielsweise in Form von Textdokumente vorliegen. Sie werden vom System nicht weiter bearbeitet.

Transformer

Die in der Analyse (siehe Abschnitt 3.4) beschriebenen Trennung der eigentlichen Informationen von den Darstellungsoptionen (siehe Contentmanagement in Abschnitt 2.1), bietet sich zur Umsetzung die Extensible Markup Language (XML) als Datenformat an (W3C 2004). XML ist ein Dokumentenformat, mit dem Formate definiert werden können (Meta-Format). Die so definierten Formate lassen sich leicht von Maschinen interpretieren und trotzdem ist es auch für Menschen lesbar. In XML wird die Struktur getrennt vom Inhalt abgelegt. Mithilfe von Extensible Stylesheet Language Transformations (XSLT) (W3C 1999) Skripten können in XML kodierte Daten in unterschiedliche Formate gewandelt werden.

Zur Transformation von XML-Dokumenten mithilfe von XSLT werden drei Komponenten benötigt (Abbildung 4.6):

1. Das XML-Dokument bildet die Datenquelle für die Transformation. Dabei wird die XML-Datei umstrukturiert beziehungsweise in ein anderes Format überführt.

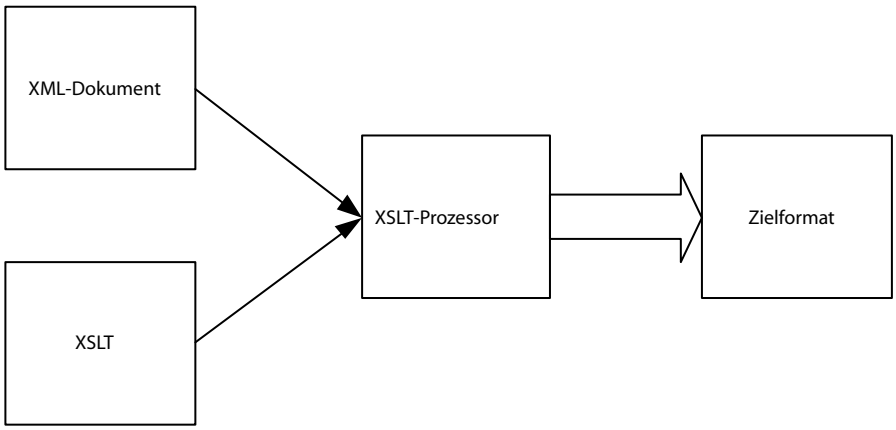

Abbildung 4.6.: Umwandlung von XML-Dokumenten

2. Das XSLT-Stylesheet beinhaltet die Transformationsregeln und schreibt vor, wie die Bestandteile der XML-Datei in das Zielformat zu überführen sind.
3. Der XSLT-Prozessor führt den eigentlichen Transformationsprozess durch. Er wendet die Regeln des Stylesheets auf die XML-Daten an und generiert das neue Dokument im Zielformat.

Replicationmanager

Wie in der Analyse im Abschnitt 3.4 angesprochen, muss beim partiellen Datenabgleich zusätzlich darauf geachtet werden, nur die aktuell relevanten Informationen auf das Gerät zu laden. Wenn eine vollständige Synchronisation nicht möglich ist, besteht die Möglichkeit nicht nur einzelne Dokumente mitzunehmen, sondern alle zu einem Dokument gehörenden Einträge gleich mit dem Dokument. Oder es werden alle im letzten Monat bearbeiteten Dokumente und zusätzlich die in den bearbeiteten Dokumenten referenzierten Dokumente mitgenommen.

Unter der Annahme, aktuell relevante Informationen sind entweder wenige Tage alt oder sind mit wenige Tage alten Einträgen verknüpft, ist ein Datenabgleich der relevanten Informationen eine kombinierte Breiten- und Tiefensuche. Wobei die Breite und Tiefe abhängig davon ist, wie viel Speicher zur Verfügung steht.

Der Abgleich von Informationen zwischen unterschiedlichen Datenbeständen soll im Rahmen dieser Arbeit nicht näher betrachtet werden, da das Thema den Rahmen dieser Arbeit sprengen würde. Ein möglicher Ansatz könnte über die Synchronization Markup Language (SyncML) erfolgen (OMA 2004). Es müssten sich keine Gedanken mehr über ein Protokoll

gemacht werden. Zusätzlich könnte das persönliche Wissensmanagementsystem auch mit anderen Systemen Daten austauschen.

Bei SyncML handelt es sich um einen Standard zum Datenabgleich, der auf einem einheitlichen und technisch ausgereiften Protokoll basiert. Darüber hinaus wird SyncML auf breiter Basis akzeptiert. Das Protokoll wurde auf Initiative führender Mobilfunkgeräte- und Computerhersteller entworfen, die sich Anfang 2000 zum SyncML-Konsortium zusammengeschlossen haben. SyncML ist zugleich Beschreibungssprache und Protokollvereinbarung. Neben der Spezifikation ist ebenfalls ein kostenloses SyncML-Reference-Toolkit erhältlich. Hiermit können Produkte auf Basis der SyncML-Regeln synchronisationsfähig gemacht werden.

4.3. Entwurf der Gesamtarchitektur

Nachdem die Anwendungslogik steht, wird nun überlegt, wie diese Logik in eine Gesamtarchitektur eingepasst werden kann. Zu Standardproblemen existieren inzwischen auch Standardlösungen. Es handelt sich hierbei um ein Baukastensystem von Konzepten, die sich auf häufig wiederkehrende Probleme in unterschiedlicher Zusammensetzung anwenden lassen.

Während sich Gamma u. a. (1994) mit ihren Entwurfsmustern die Zusammenhänge zwischen Klassen beschreiben, betrachten Buschmann u. a. (1996) mit ihren Architekturmustern vollständige Systemarchitekturen.

Schichtenarchitektur

Benutzungsoberflächen und Datenhaltung ändern sich aufgrund des technischen Fortschritts schneller, als das Fachkonzept. Durch die Entkopplung der verschiedenen Bereiche kann zum Beispiel das Fachkonzept unverändert weiterbenutzt werden, obwohl die Benutzungsoberfläche komplett neu gestaltet wurde. Es kann auch zur Steigerung der Leistungsfähigkeit des Systems die Datenbank gewechselt werden, ohne dass dies Auswirkungen auf die übrigen Bereiche der Anwendung hat.

Aus den oben genannten Gründen soll auch hier eine klare Trennung der unterschiedlichen Bereiche des Systems vorgenommen werden.

Informationssysteme können in die folgenden Bereiche unterteilt werden: Präsentation, Fachdomäne und Infrastruktur/ Datenhaltung (Buschmann u. a. 1996, Seite 47). Diese Unterteilung in eine Schichtenarchitektur stellt ein Grundprinzip der Entwicklung dar (Parnas

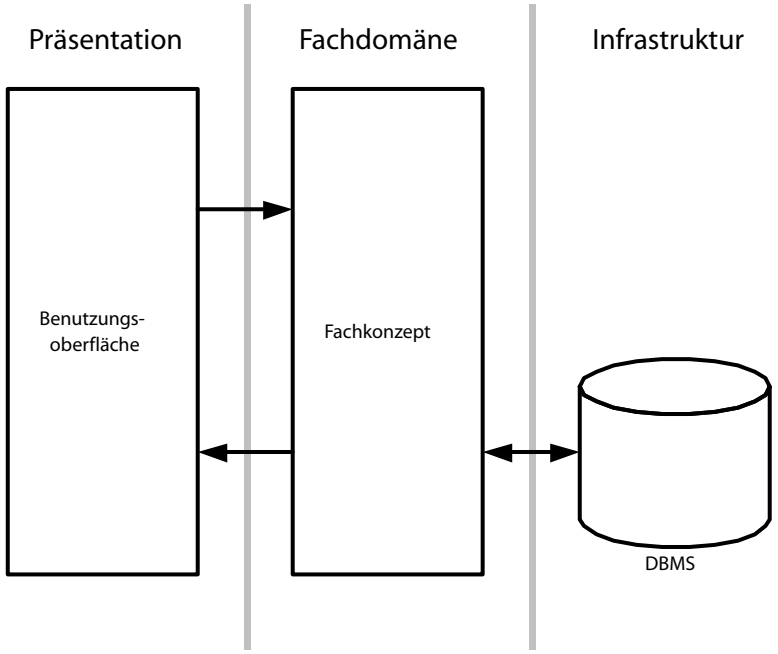

Abbildung 4.7.: Schichtenarchitektur

1978). Teilweise wird noch die Datenhaltung durch eine zusätzliche Zugriffsschicht gekapselt. Klassen interagieren jeweils nur mit den Klassen aus derselben oder aus einer angrenzenden Schicht. Die Interaktion zwischen den verschiedenen Schichten erfolgt ausschließlich über klar definierte Schnittstellen der Schichten nach außen. Die Unterteilung in Schichten stellt eine Form des Geheimnisprinzips da.

Model/View/Controller (MVC) Architektur

Wie wir oben gesehen haben, ist die Entkopplung von Benutzungsoberfläche und Fachkonzept von elementarer Bedeutung um eine Kontinuität im Fachkonzept zu gewährleisten. Die in dem Konzept der Schichtenarchitektur vorgesehene Trennung von Präsentation und Fachdomäne soll nun umgesetzt werden. Der klassische Ansatz zur Entkopplung ist das folgende Paradigma.

Entkoppelung von Präsentation, Ablaufsteuerung und Modell/ Datenhaltung eines Systems, wie in der Schichtenarchitektur vorgesehen, wird im Model/ View/ Controller (MVC) Paradigma umgesetzt. Ziel ist es, die Flexibilität zu erhöhen und die Komplexität zu reduzieren. Das MVC-Paradigma hat seinen Ursprung in Smalltalk-76, dem Vorläufer von Smalltalk-80 (Reenskaug u. a. 1995, Seite 318). Es wurde am Xerox Palo Alto Research Center (PARC)

entwickelt, um die Benutzerinteraktion der ersten Computer mit grafischer Benutzungsoberfläche zu verwalten[1] (Reenskaug 1979a, b).

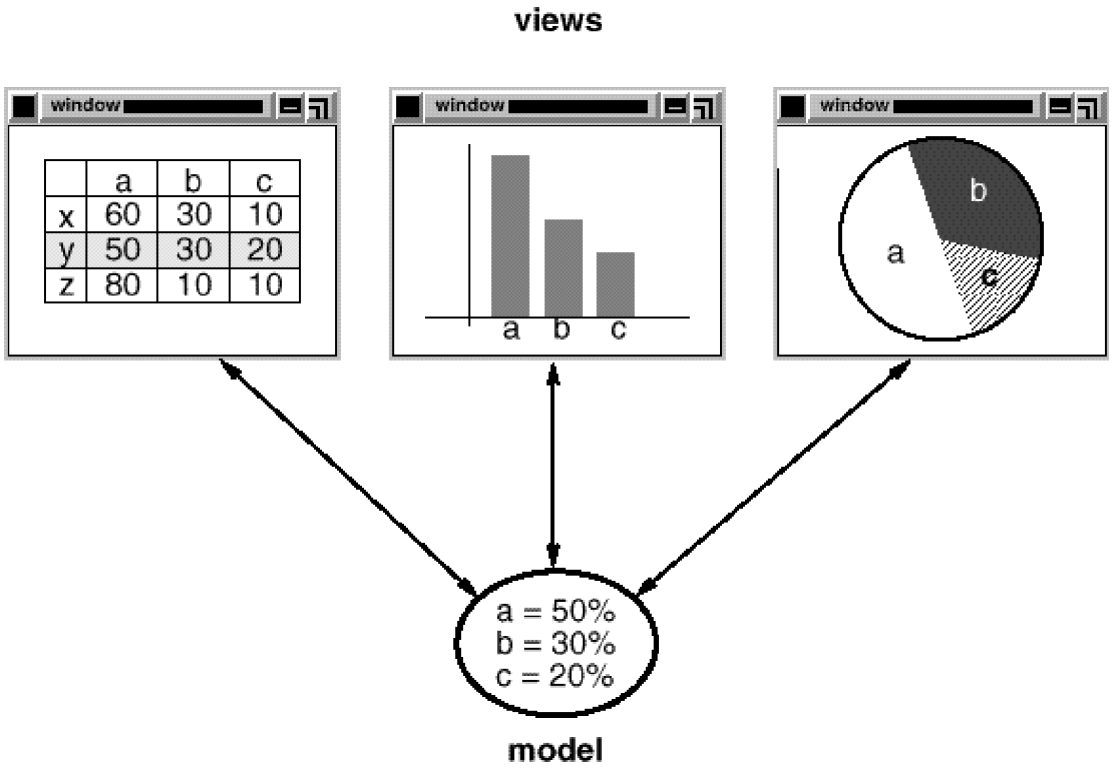

Abbildung 4.8.: Beziehung von Model und Views (nach Gamma u. a. 1994)

Das klassische Beispiel zur Erklärung des MVC-Paradigmas stammt aus dem Entwurfsmusterbuch (Gamma u. a. 1994). Eine Anwendung soll mehrere verschiedene Sichten auf dieselben Daten bereitstellen. Im Beispiel, wie in Abbildung 4.8 zu erkennen, eine Kalkulationstabelle, ein Balkendiagramm und ein Tortendiagramm. Jede dieser Sichten soll zur selben Zeit auch von unterschiedlichen Benutzern betrachtet werden können. Die Anwendung muss gewährleisten, dass die verschiedenen Sichten bei Änderungen der zugrunde liegenden Daten oder des Modells aktualisiert werden. Um das Modell zu verändern, muss eine Änderungsanforderung an den Controller gesendet werden. Nach einer Änderung des Modells müssen alle Sichten aktualisiert werden.

In Abbildung 4.9 ist die generelle Struktur der MVC-Architektur dargestellt. Die einzelnen Bestandteile sollen kurz beschrieben werden:

[1] Eine weitere Erfindung von PARC, die nicht von Xerox kommerziell verwertet wurde, neben dem Konzept des Notebooks, der graphischen Benutzungsoberfläche, . . . und Ubiquitous Computing

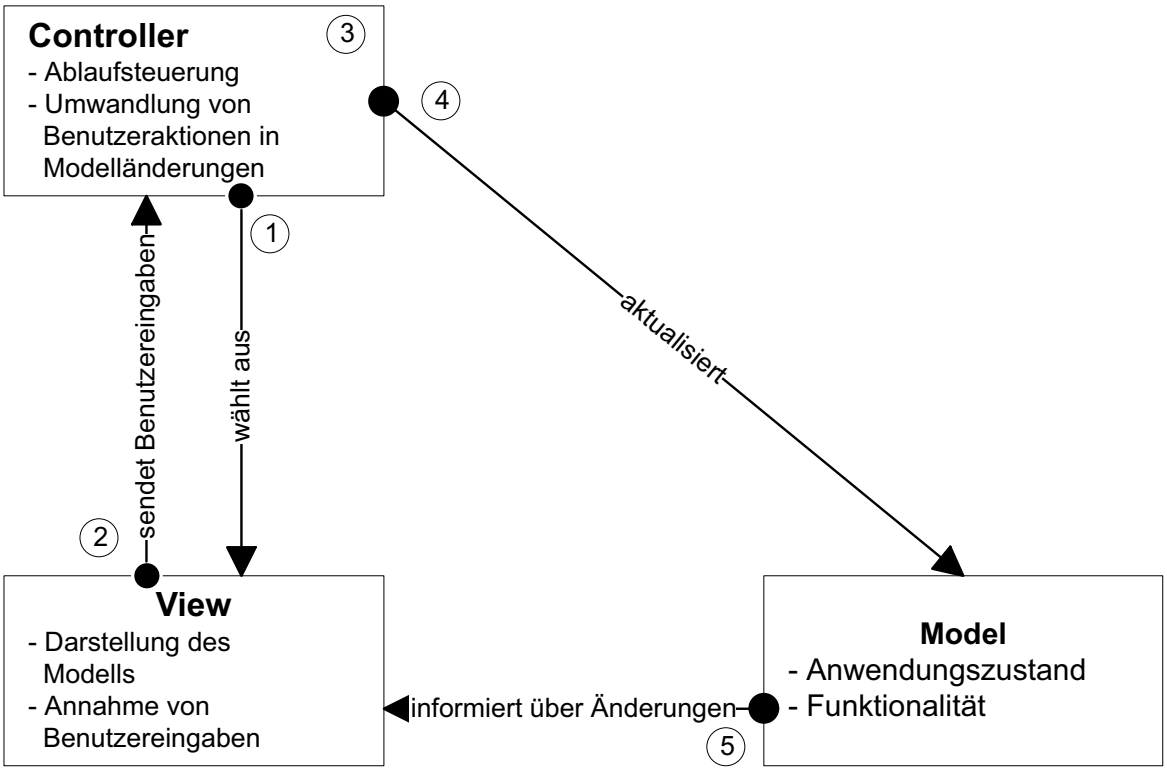

Abbildung 4.9.: Model/ View/ Controller (MVC) Architektur

Model Das Model beinhaltet die fachlichen Aspekte der Anwendung, die in Abbildung 4.3 dargestellte Anwendungslogik. Es kapselt die Daten und den Zustand der Anwendung. (Und informiert die Views über Änderungen.)

View Eine View ist eine visuelle Repräsentation des zugrunde liegenden Modells. Des Weiteren werden Eingaben vom Benutzer entgegen genommen und an den Controller weitergeleitet.

Controller Mittels des Controllers wird das Verhalten der Anwendung gesteuert. Der Controller legt fest, welche Funktionalität des Modells aufgrund einer Benutzereingabe auszuführen ist. Auch wählt der Controller die zur Darstellung des Modells nötige View aus.

Der typische Ablauf sieht wie folgt aus:

1. Aufgrund einer Benutzeranfrage wählt die Ablaufsteuerung die zur Anfrage passende Sicht aus und leitet den Benutzer dorthin. Die angeforderten Daten werden angezeigt.
2. Benutzereingaben werden an die Ablaufsteuerung weitergereicht.

3. Die Ablaufsteuerung überprüft die eingegebenen Daten auf semantische und syntaktische Korrektheit.
4. Fehlerfreie Eingaben werden in Anweisungen zur Modelländerung umgesetzt.
5. Immer, wenn Veränderungen am Modell vorgenommen werden, müssen auch die Sichten auf dieses Modell aktualisiert werden.

Die automatische Benachrichtigung der abhängigen Sichten, bei einer Zustandsänderung des Modells, wird meist über das Beobachtermuster (Observer) (Gamma u. a. 1994, Seite 293) realisiert.

Anwendung der Model/ View/ Controller (MVC) Architektur

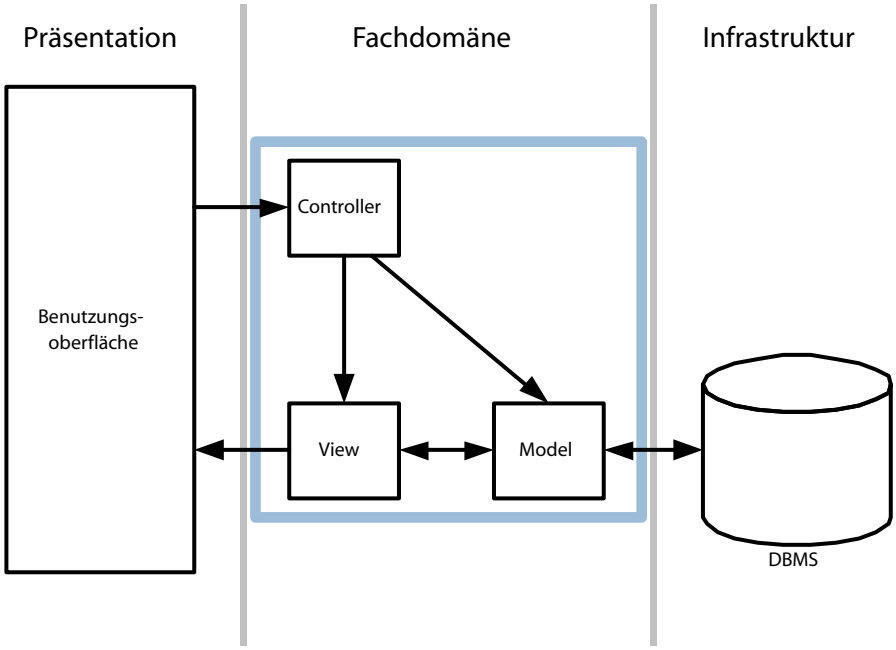

Abbildung 4.10.: Schichtenarchitektur mit Model/ View/ Controller

Abbildung 4.10 erweitert die im Abschnitt 4.3 eingeführte Schichtenarchitektur um das MVC-Paradigma. Die im Abschnitt 4.2 entwickelte Anwendungslogik zur Umsetzung der domänenspezifischen Aufgaben befindet sich in der Klasse „Model“.

Objektpersistenz

Nachdem die Trennung von Präsentation und Fachdomäne umgesetzt ist, soll nun die Infrastrukturschicht betrachtet werden. In dieser Schicht ist hauptsächlich die Datenhaltung angesiedelt. Bevor die Schnittstelle zur Datenhaltung, die so genannte Persistenzschicht, betrachtet wird, sollen zuvor noch grundlegende Dinge zur Persistenz in objektorientierten Systemen behandelt werden.

Objekte einer objektorientierten Anwendung sind „transient“. Sie gehen bei jedem Programmende verloren. Jede ernsthafte Anwendung erfordert die dauerhafte Speicherung von Objekten, so genannte „persistente“ Objekte (Atkinson u. a. 1982; Atkinson 1991).

Die am weitesten gehende Form der dauerhaften Speicherung von Daten oder Objekten ist die orthogonale Persistenz. Atkinson und Morrison (1995) beschreiben die Eigenschaften orthogonal persistenter Systeme anhand der folgenden Prinzipien:

Persistenzunabhängigkeit Es müssen keine expliziten Transfers vom flüchtigen (transienten) zum dauerhaften (persistenten) Zustand durchgeführt werden.

Typ-Orthogonalität Datenobjekte beliebigen Typs können dauerhaft gespeichert werden.

Persistenz durch Erreichbarkeit Persistenz ist über die Erreichbarkeit eines Objektes von einer oder mehreren persistenten Wurzeln aus definiert.

Eine schwächere Form stellt die transparente Persistenz da. In diesem Fall wird die Notwendigkeit von zusätzlichem Programmcode oder Metadaten akzeptiert.

Die unterschiedlichen Ansätze zur dauerhaften Speicherung von Objekten sollen im Folgenden diskutiert werden.

Im Vorgriff auf das folgende Kapitel sollen hier bereits Komponentenentscheidungen behandelt werden. Die Gesamtarchitektur variiert je nach verwendeter Persistenztechnik. Da keine proprietäre Software verwendet werden soll, schränkt sich die Auswahl möglicher Komponenten zusätzlich ein. Daher soll bereits hier kurz auf Basis der Komponenten argumentiert werden.

Objektserialisierung

Objektserialisierung stellt die einfachste Persistenztechnik dar. Objekte werden in einen Bytestrom umgewandelt und dieser dann als Datei abgespeichert. Bei der Umwandlung werden nacheinander die Signaturen der Klassen zusammen mit den Werten der Klassenattribute geschrieben. Diese Technik ist in Java bereits integriert. Sie wird beispielsweise bei Javas RMI verwendet. In der Anwendung ist die Serialisierung jedoch umständlich, da Klassen von

Hand serialisiert/ deserialisiert werden müssen. Auch ist inkrementelles Laden nicht möglich. Objekte müssen vollständig im Hauptspeicher deserialisiert werden, um mit ihnen arbeiten zu können. Da der Bytestrom unstrukturiert ist, eignet er sich auch nicht zum Durchsuchen. Hierbei müssen alle Daten in den Arbeitsspeicher geladen werden. Daher sollte die Serialisierung nur für kleine Datenbestände benutzt werden. Hinzukommt, dass die Objektserialisierung nicht transparent ist. Bei der Entwicklung müssen die Objekte explizit persistent gemacht werden. Auch um den Schutz der Daten muss sich der Entwickler selbst kümmern.

Objektorientierte Datenbanksysteme (OODMBS)

Anstatt selbst die Objekte „flach zu klopfen“, kann diese Aufgabe auch einem System überlassen werden, welches speziell auf das Speichern und Wiederauffinden von Objekten ausgelegt ist. Objektorientierte Datenbankmanagementsysteme (OODMBS) wandeln Objekte in flache Strukturen und legen diese unter Verwendung von bekannten Datenbankindexmechanismen direkt auf dem Massenspeicher ab. Mittels OODBMS kann eine nahtlose Verbindung zwischen objektorientierter Anwendung und Datenhaltung hergestellt werden.

Bisher konnten sich jedoch objektorientierte Datenbankmanagementsysteme nicht durchsetzen. Auch waren dem Autor zum Zeitpunkt der Ausarbeitung keine frei verfügbaren OODBMS bekannt.

Exkurs: XML-Datenbanken

Da hier bereits vorgesehen ist, die nicht-Kalenderobjekte im XML-Format abzulegen, um sie einfach in andere Ausgabeformate umwandeln zu können, liegt die Überlegung nahe, sämtliche Informationen in XML abzulegen und eine native XML-Datenbank zu verwenden. Daher soll diese neue Form der Datenbanken hier kurz betrachtet werden.

XML ist ursprünglich als Datenaustauschformat gestartet. Zur anwendungsinternen Kommunikation sollte XML daher eigentlich nicht verwendet werden. Es ist durch das universelle Datenformat zusätzlicher Aufwand nötig, da jedes Datum näher spezifiziert werden muss. Innerhalb einer Anwendung können die Vorteile eines universellen Formats auch nicht genutzt werden. Mit der zunehmenden Verbreitung von XML zum Datenaustausch zeichnet sich jedoch eine Tendenz ab, Daten auch direkt im XML-Format zu speichern. So existieren inzwischen einige XML-Datenbanken, von denen einige frei sind (Xindice 2004; eXist 2004). Hier können XML-Daten ohne Umwandlung direkt abgelegt werden. So kann mit beliebigen, zur Entwicklungszeit noch nicht notwendigerweise bekannten XML-Daten gearbeitet werden.

Native XML-Datenbanken weisen aufgrund ihrer recht jungen Entwicklungsgeschichte keine stabile Schnittstelle für den Zugriff auf die Daten auf. Die Standardisierung befindet sich am

Anfang (XMLDB 2004). Daher soll hier davon abgesehen werden, sie zum jetzigen Zeitpunkt zu nutzen.

Relationale Datenbanksysteme (RDBMS)

> „A hundred years from now, I'm quite sure, database systems will still be based on Codd's relational foundation." (Date 2000)

Übrig bleiben die erprobten relationalen Datenbankmanagementsysteme (RDBMS). Diese haben sich über lange Zeit bewährt und sind auch in ausgereifter Form frei verfügbar. Es besteht jedoch eine konzeptionelle Kluft zwischen der objektorientierten- und der relationalen Welt.

Objektrelationale Abbildung

Die grundlegende Problematik der konzeptionellen Kluft wird als „Impedance Mismatch"[2] bezeichnet. Sie entsteht durch die fehlende Abbildung von objektorientierten Strukturen auf das Datenmodell relationaler Datenbankmanagementsysteme (RDBMS). Der Begriff entstammt der Elektrotechnik und bezeichnet dort das Problem von zwei miteinander verbundenen Koaxialleitungen mit unterschiedlichem Scheinwiderstand. Ähnlich wie die beiden Koaxialleitungen aneinander angepasst werden müssen, muss auch zwischen den unterschiedlichen Welten von Objekten und Relationen eine Brücke gebaut werden.

Es existieren Strategien, um die konzeptionelle Kluft zwischen der objektorientierten und der relationalen Welt zu überbrücken. Mithilfe dieser Strategien können zwar nicht alle Fälle abgedeckt werden, in der Praxis haben sie sich jedoch als ausreichend erwiesen (Brown und Whitenack 1996; Ambler 1998).

Da in gleicher Weise mit transienten und persistenten Objekten gearbeitet werden soll, scheidet die manuelle objektrelationale Abbildung aus. Eine transparente Persistenz ist hier nicht gegeben. Objekte müssen eigenständig dafür sorgen, dass sie per JDBC persistent gehalten werden. Daher werden Werkzeuge zur objektrelationalen Abbildung, auch O/R-Mapper genannt, eingesetzt.

Umsetzung der Objektpersistenz

Die in Abbildung 4.11 hinzugefügte Persistenzschicht verbindet das objektorientierte Fachmodell mit Techniken zur dauerhaften Speicherung der Objekte. Durch diese Schnittstelle

[2] engl. Impedanzfehlanpassung

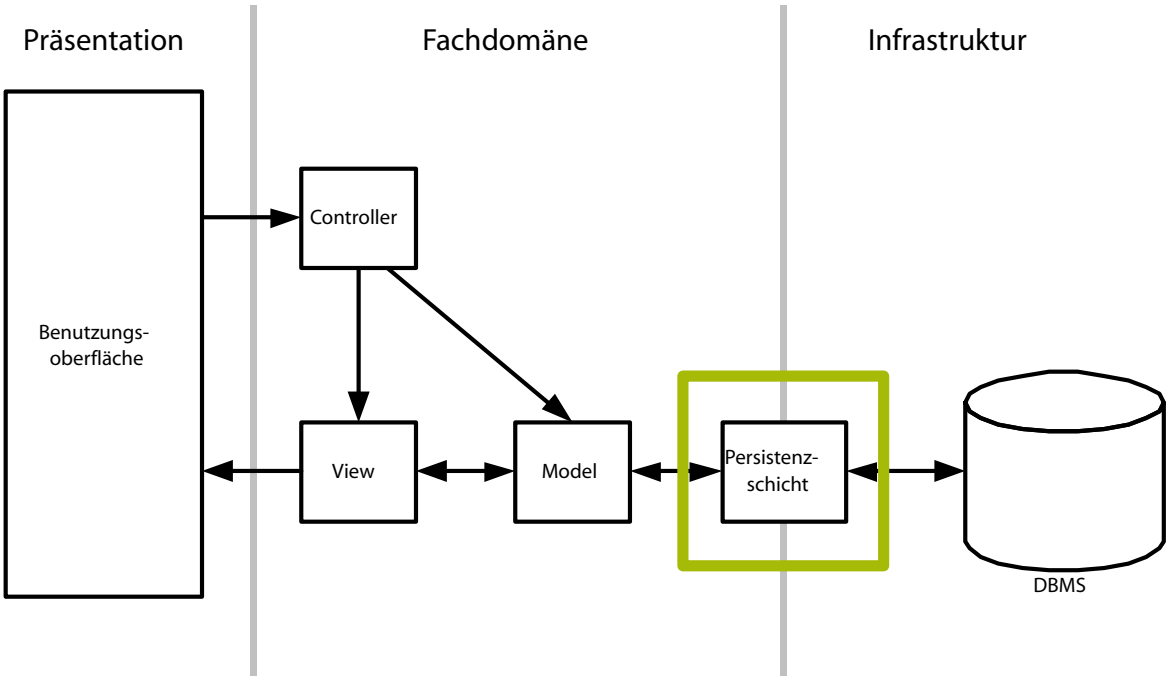

Abbildung 4.11.: Schichten-Architektur mit Model/ View/ Controller und Persistenzschicht

ist es für das Fachmodell unerheblich, ob eine objektorientierte oder eine relationale Datenbank verwendet wird. Mit der Schnittstelle zur Datenhaltung und der daran angeschlossenen Datenhaltung selbst ist das System vollständig.

4.4. Zusammenfassung

Informationen werden im Kontext gespeichert. Dies wird dadurch erreicht, dass jedes Informationsobjekt mit den Informationsobjekten verknüpft werden kann, mit denen es in einer Beziehung steht. Die Verknüpfungen erfolgen ausschließlich mittels syntaktischer Verweise. Auf diese Weise muss kein Aufwand betrieben werden, um beispielsweise ein Modell für semantische Verknüpfungen zu erstellen.

Die Architektur (vgl. Abb. 4.11) folgt der klassischen Schichteneinteilung: Präsentation, Fachdomäne und Infrastruktur. Zur Entkoppelung der Benutzungsoberfläche wird das Model/ View/ Controller-Paradigma verwendet. Die Datenhaltung wird mittels einer Persistenzschicht von der Anwendungslogik getrennt.

5. Realisierung

Der ausgearbeitete Entwurf wird mithilfe von Standardkomponenten umgesetzt. Hierzu wird zunächst eine geeignete Plattform ausgewählt. Auf Basis dieser Entscheidung werden dann die zur Umsetzung der geforderten Funktionalitäten benötigten Komponenten ausgewählt.

5.1. Desktop- versus Client/ Server-Anwendung

Ein entscheidender Punkt bei der Umsetzung des Entwurfs ist die Schaffung von allgegenwärtigen Informationen. Wie bereits mehrfach erwähnt ist dies nur möglich, wenn für die unterschiedlichen Einsatzzwecke darauf angepasste Geräte eingesetzt werden. Um den Aufwand im ersten Schritt überschaubar zu halten, wird das Programm zunächst zentral auf einem Server installiert und die anderen Geräte können ausschließlich über den Webbrowser zugreifen. Die Einfachheit dieser Teillösung beinhaltet einen geringeren Komfort für den Benutzer, da die Interaktion mit dem System eingeschränkt ist. Ein Internetbrowser ist inzwischen auf nahezu jedem in Frage kommenden Gerät vorhanden. Somit braucht die Clientseite nicht weiter betrachtet werden. Nichtsdestotrotz ist die Systemarchitektur auch für den Einsatz auf allein operierende Geräte konzipiert.

Webanwendung

Anwendungen, deren Benutzungsoberfläche über das World Wide Web (WWW) bereitgestellt werden, werden ganz allgemein als Webanwendungen bezeichnet. Es handelt sich dabei um verteilte Anwendungen. Es können sowohl Komponenten im Internetbrowser, als auch auf Servern verwendet werden. Die Kommunikation zwischen der jeweiligen Anwendung und ihrem Client findet über das Hypertext Transfer Protocol (HTTP) statt.

Bei Webanwendungen müssen prinzipbedingte Eigenarten beachtet werden. So muss beispielsweise beachtet werden, dass Benutzer in den Browsern Aktionen ausführen können, die Einfluss auf die als nächste auszuführende Funktion haben, jedoch von der Anwendung lediglich festgestellt und nicht kontrolliert werden können. Beispiele hierfür sind das Öffnen

mehrerer Browserfenster oder das Zurückspringen innerhalb des Browsers. Dieses Benutzerverhalten muss bei der Erstellung der Anwendung unbedingt berücksichtigt werden.

Neben den Eigenarten, die durch das Client/ Server-Prinzip bedingt sind, existieren bei Webanwendungen noch weitere Eigenarten. So die durch das verwendete Übertragungsprotokoll HTTP. Das Protokoll ist zustandslos. HTTP-Server verwalten keine Informationen über ihre Clients. Jede HTTP-Anfrage wird gleich behandelt, unabhängig davon, was in der vorhergehenden Anfrage war. Sollen Operationen in mehreren Schritten ausgeführt werden. Und ist das Ergebnis des vorhergehenden Schritts entscheidend für die Bearbeitung des aktuellen Schritts, so müssen die auf HTTP aufsetzenden Anwendungen selbst einen Mechanismus implementieren, um sich Zustände zu merken.

Wie bereits erwähnt, liegt ein entscheidender Vorteil von Webanwendungen gegenüber anderen Anwendungssarten in ihrer Verfügbarkeit. Sie müssen nicht auf den jeweiligen Geräten installiert werden, sondern sind mit der Installation auf einem Web-Server sofort von jedem internetfähigen Gerät aus nutzbar.

Ein Nachteil von Webanwendung ist sicherlich der momentan noch eingeschränkte Einsatzbereich. Aufgrund der, im Vergleich zu Desktop-Anwendungen funktionsarmen Browser, lassen sich rechen- oder grafikintensive Anwendungen nicht mit ihnen realisieren. Dies müsste dann auf dem Server erfolgen. Der Browser muss die auf dem Server erstellen Grafiken nur noch darstellen. Negativ wirkt sich jedoch die eingeschränkte Interaktionsfähigkeit aus. Eingaben können auf dem Client höchstens grob auf offensichtliche Fehler überprüft werden. Eine vollständige Prüfung kann erst erfolgen, nachdem alle Daten zum Server gesendet wurden.

5.2. Plattform

Zurzeit werden im Bereich der verteilten Anwendungen hauptsächlich zwei Plattformen verwendet: Microsofts .NET und Sun Microsystems J2EE Ansatz. Diese sollen nun gegenübergestellt werden, um dann eine auszuwählen.

J2EE

Java 2 Enterprise Edition (J2EE) ist eine Spezifikation, welche Standards zur Implementierung, Konfiguration, Verteilung und zum Einsatz von unternehmensweiten Anwendungen definiert. Es wird hierbei ausschließlich Java als Programmiersprache verwendet. Die Spezifikationen werden hierbei vom Java Community Process (JCP) erstellt. Dem JCP gehören

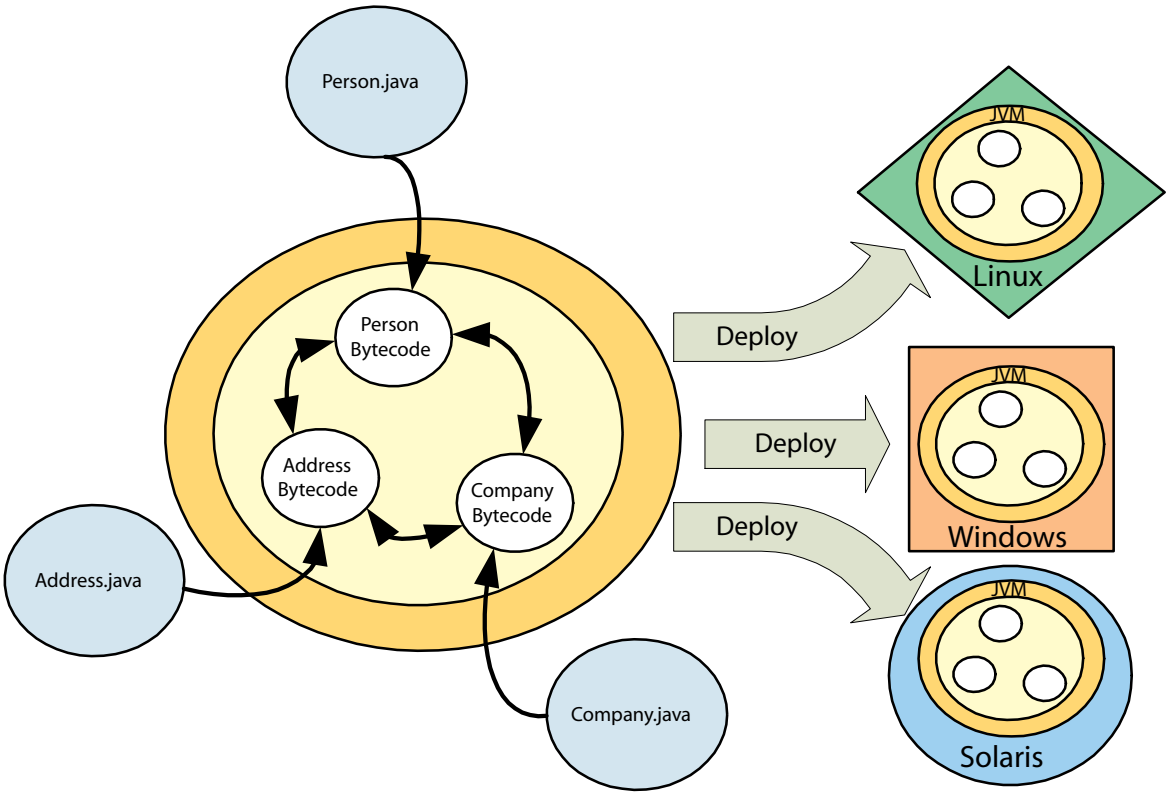

Abbildung 5.1.: J2EE-Plattform (nach Williams 2003, Seite 61)

namhafte Firmen wie BEA, IBM und Oracle an. Ein Produkt, welches J2EE-lizenziert werden soll, muss hierfür die Compatibilty Test Suite (CTS) erfolgreich durchlaufen. Auf Grundlage der J2EE-Spezifikation werden von unterschiedlichen Herstellern Produkte angeboten. Allerdings wurden Unklarheiten in der Spezifikation von den verschiedenen Herstellern unterschiedlich interpretiert. Aufgrund der Interpretationsmöglichkeiten der Spezifikation sind Programme nicht immer ohne Anpassungen auf Produkten unterschiedlicher Hersteller nutzbar.

Bei Java wird der Quellcode in ein Zwischenformat, den Bytecode, kompiliert. Der Bytecode wird innerhalb einer Laufzeitumgebung ausgeführt (siehe Abbildung 5.1). Es existieren für die unterschiedlichsten Betriebssysteme Javalaufzeitumgebungen.

.NET

Während J2EE eine reine Spezifikation ist, so umfasst .NET zusätzlich Produkte, die auf dieser Spezifikation beruhen. Alle .NET-Applikationen nutzen dieselbe Laufzeitumgebung,

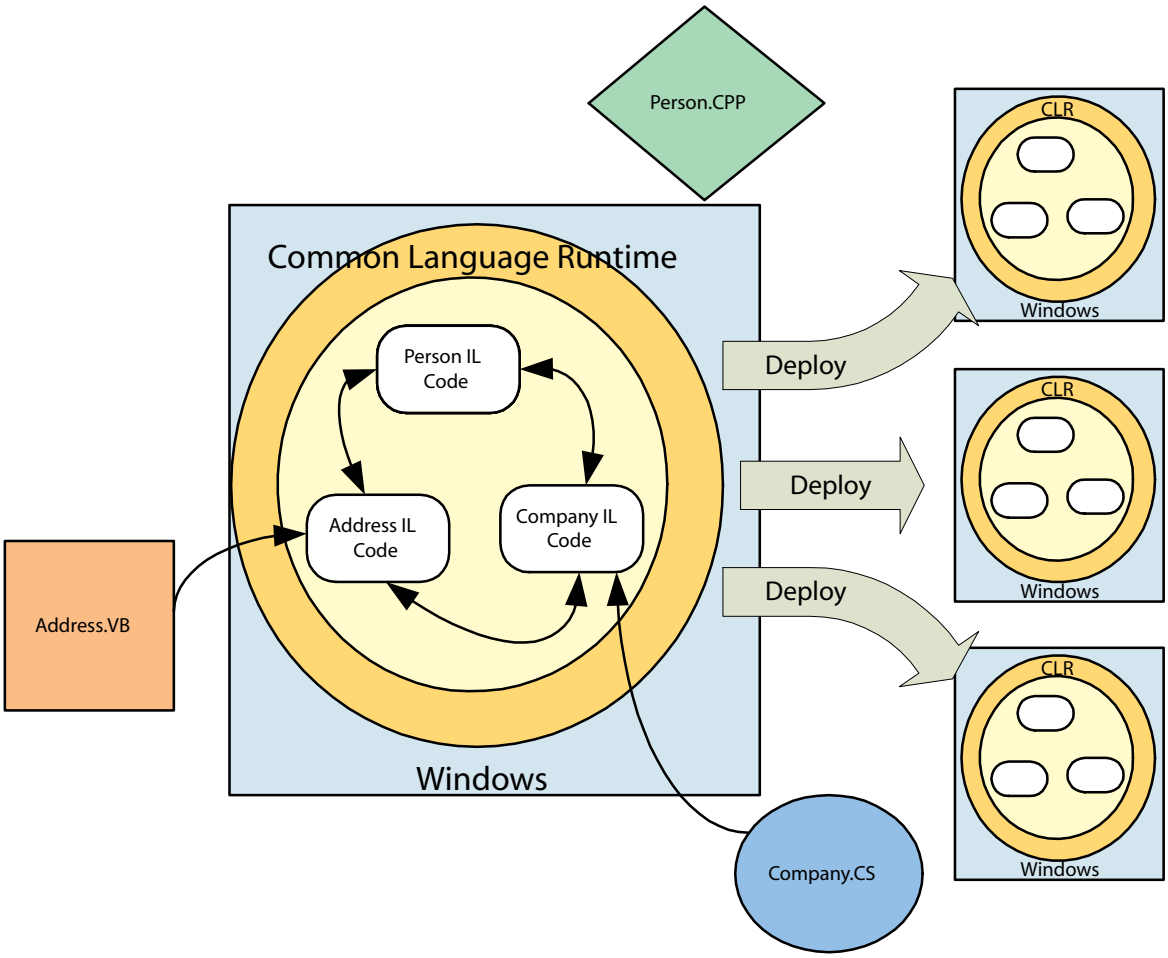

Abbildung 5.2.: .NET-Plattform (nach Williams 2003, Seite 60)

die Common Language Runtime. Abbildung 5.2 soll den .NET-Ansatz deutlich machen. Über diese ist es möglich, verschiedene Programmiersprachen innerhalb von .NET zu verwenden. Die Compiler erzeugen eine Zwischensprache, die so genannte Intermediate Language (IL). Durch diesen Zwischenschritt sind die Sprachen interoperabel. Der Zwischencode wird erst vor der Ausführung durch den Just-in-Time-Compiler in ausführbaren Maschinencode übersetzt. Um zu erreichen, dass verschiedene Sprachen wirklich denselben Code erzeugen, muss der IL-Code bestimmten Bedingungen genügen. Dazu gehören identische Datentypen. Diese liefert unter .NET das Common Type System (CTS). Ein Integer in C# und einer in C++ ist somit trotz unterschiedlicher Syntax derselbe Datentyp. Dieser Umstand schränkt die programmiersprachenunabhängigkeit allerdings erheblich ein. Denn Datentypen müssen im erzeugten Bytecode übereinstimmen.

Von Haus aus ist .NET auf die Windows-Plattform beschränkt. Alle Komponenten kommen aus einer Hand, von einem einzigen Hersteller. Es existieren jedoch inzwischen An-

sätze, das .NET-Framework auch auf anderen Betriebssystemen nutzbar zu machen. So realisiert beispielsweise das Mono Projekt (de Icaza 2004) eine freie Implementierung des .NET-Frameworks, die derzeitig unter Linux, Solaris, Mac OS X und einigen anderen Unix-Derivaten sowie, als Alternative zu Microsofts .NET-Framework, auch unter Windows 2000 und XP funktioniert.

J2EE versus .NET

Im Rahmen einer Auseinandersetzung über die geeignete Plattform für Web Services hat sich in den Communications of the ACM je ein Vertreter von Sun Microsystems (Williams 2003) und von Microsoft (Miller 2003) über die Vor- und Nachteile von J2EE und .NET geäußert. Im Prinzip kommen beide zu einem ähnlichen Urteil: Viele Argumente für oder gegen eine der beiden Plattformen können nur als FUD – fear, uncertainty and doubt [1] – bezeichnet werden. Wenn nicht gerade das Hauptunterscheidungsmerkmal „Plattformunabhängigkeit versus Sprachenunabhängigkeit“ ausschlaggebend ist, kommt die Entscheidung vor allem auf strategische Dinge an. Möchte man sich zum Beispiel an einen Hersteller binden? Auch die Herstellerunabhängigkeit kann Probleme bereiten. Denn so müssen gegebenenfalls die Produkte von unterschiedlichen Herstellern aufwendig aufeinander angepasst werden.

Beim Vergleich der Java-Umgebung mit dem .NET Framework fallen die folgende Punkte auf: In Java sowie auch im .NET Framework wird der Quellcode in ein Zwischenformat kompiliert und der Code wird innerhalb einer Laufzeitumgebung ausgeführt. Während Java plattformunabhängiger ist, können bei .NET unterschiedliche Programmiersprachen eingesetzt werden. Während Java inzwischen sehr ausgereift ist, bietet .NET eine komfortablere Unterstützung von neueren Techniken, wie zum Beispiel Web Services.

Das Bundesinnenministerium hat im Rahmen der Initiative BundOnline 2005 Standards und Architekturen für E-Government-Anwendungen miteinander verglichen. Dort wird .NET als „unter Beobachtung“ eingestuft, da es nicht betriebssystemunabhängig ist und nicht auf offenen Standards aufbaut (KBSt 2003, Seite 78). Währenddessen wird Java und J2EE als Basistechnologie ausgewählt.

„Die Entscheidung für den Einsatz von Java basiert auf der Plattformunabhängigkeit, der optimalen Unterstützung von objektorientierten Software-Techniken, der Stabilität der Ausführungsumgebung und der großen Anzahl der frei oder auch kommerziell verfügbaren APIs“ (KBSt 2003, Seite 62). Den Ausführungen wird sich hier angeschlossen und die Java-Plattform ausgewählt.

[1] „FUD“ steht für „Fear, Uncertainty and Doubt“ – „Angst, Unsicherheit und Zweifel“. Mit diesem Akronym charakterisierte seinerzeit Gene Amdahl, Chefentwickler von IBMs 370 Mainframe, die Taktik von IBM, die Konkurrenz mit Gerüchten und halben Ankündigungen zu verunsichern.

5.3. Lizenzmodell

Freie Software ist eine interessante Alternative zu herkömmlicher, proprietärer Software. Aufgrund der Tatsache, dass die Quelltexte von freien Softwareprodukten offen einsehbar sind, wird für derartige Software häufig die englische Bezeichnung Open Source Software (OSS) verwendet. In den vergangenen Jahren wurde dieses Phänomen immer populärer. Der breite Einsatz des Apache HTTP-Servers, der seit Jahren der am häufigsten eingesetzte Webserver ist (Netcraft 2003), demonstriert eindrucksvoll, dass freie Software mindestens so gut, wenn nicht sogar besser ist, als kommerzielle Software. Die im Abschnitt 3.4 aufgestellten hohen Anforderungen, bei der gleichzeitigen Forderung nach möglichst geringen Kosten, lassen sich am besten mit freien, quelloffenen Komponenten umsetzen.

Eine Herausforderung bei der Auswahl von Open Source Komponenten stellt häufig die Frage dar, ob das Projekt überleben wird. Einen guten Anhaltspunkt stellt die Größe der Entwickler- und der Anwendergemeinde dar. Vor der Auswahl sollte auch die Aktivität der Entwicklergemeinde an der Komponente betrachtet werden. Vielleicht wird das Projekt bereits nicht mehr aktiv weiterentwickelt? Ist eine Komponente bereits in vielen Projekten im Einsatz, so verringert sich die Gefahr, dass die Weiterentwicklung der Komponente auf kein Interesse mehr stößt und eingestellt wird. Bei der Wahl eines Exoten kann es ebenfalls schnell passieren, dass man ungewollt als Pilotanwender dient. Häufig ist die Dokumentation, beziehungsweise ihre Abwesenheit, ein weiteres Problem bei freier Software.

5.4. Komponenten

Wie im Abschnitt 5.3 beschrieben, muss bei der Auswahl von Komponenten aus dem Bereich der freien Software einiges beachtet werden. Nun sollen nun Komponenten zur Umsetzung der Architektur ausgewählt werden.

Web Frameworks

Bei Webanwendungen gibt es viele Aufgaben, die bei jeder Webanwendung gelöst werden müssen. Derartige Standardlösungen sind in so genannten Frameworks zusammengefasst.

Eine der wichtigsten Arten der Wiederverwendung, ist die Wiederverwendung von Systementwürfen. Frameworks setzen diese Art der Wiederverwendung um. „Ein Framework be-

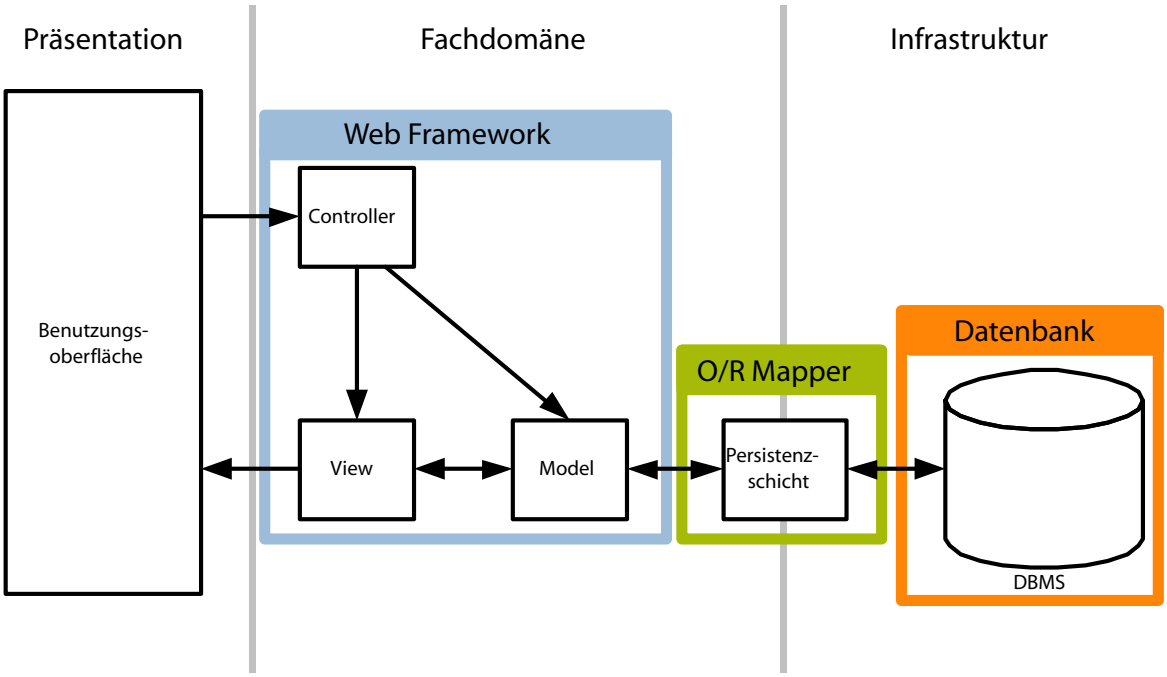

Abbildung 5.3.: Komponenten

steht aus einer Menge von zusammenhängenden Klassen, die einen wiederverwendbaren Entwurf für eine bestimmte Klasse von Software darstellen (Gamma u. a. 1994).“[2]

Wie im Entwurf im Abschnitt 4.3 beschrieben, wird hier das MVC-Paradigma zur Entkoppelung von Präsentation, Ablaufsteuerung und Modell angewendet. Es wird im Rahmen der Blueprints für die Entwicklung mit der Java 2 Plattform Enterprise Edition (J2EE) von Sun Microsystems empfohlen (Sun 2002).

Im Allgemeinen stellen Java Beans das Model dar. Für die visuelle Repräsentation des zugrunde liegenden Modells, die View, empfiehlt Sun den Einsatz von Java Server Pages (JSP). Die anzuzeigenden Daten werden aus dem Model geholt und anstelle von speziellen Tags in die HTML-Seite eingefügt. Der Controller wird nach der Empfehlung von Sun durch ein Servlet repräsentiert. Der Controller legt fest, welche Funktionalität des Modells aufgrund einer Benutzereingabe auszuführen ist. Auch wählt der Controller die zur Darstellung des Modells nötige View aus.

Bei großen Anwendungen kann das Controller-Servlet schnell unübersichtlich und dadurch schwer wartbar werden. Um dieses Problem zu umgehen, wird teilweise eine weitere Unterteilung der Komponenten vorgenommen. Die Aufteilung erfolgt hierarchisch und stellt sich als

[2] "A framework is a set of cooperating classes that make up a reusable design for a specific class of software."

baumähnliche Verknüpfung von Controller-Komponenten dar. Jeder Controller hat zwar seine eigenen Views und Models, reicht aber nicht bekannte Ereignisse an die übergeordnete Ebene weiter.

Es existieren eine Vielzahl an Java Frameworks zur Entwicklung von Webanwendungen. Sie setzen neben dem MVC-Paradigma verschiedene weitere Entwurfsmuster um und bieten mehr oder weniger Unterstützung für immer wiederkehrende Aufgaben im Problemumfeld.

Wie im Abschnitt 5.3 erläutert, sollen hier frei verfügbare Open Source Komponenten eingesetzt werden. Für Java gibt es eine Fülle von freien Web Frameworks zur Umsetzung des MVC-Paradigmas. Wird Wert auf Stabilität und Verbreitung gelegt, so bleiben nur einige wenige Frameworks übrig. In Tabelle 5.1 ist eine kleine Auswahl an freien Java-Frameworks aufgeführt.

„freshmeat“ ist nach eigenen Angaben der größte Index für Unix und plattformübergreifende Software (freshmeat 2004). In der entsprechenden Kategorie wurden alle in Frage kommenden Java-Frameworks herausgesucht. Die vier Frameworks, auf dessen Website die meisten Verweise zeigen, wurden ausgewählt.

Name	Homepage	Bemerkungen
Struts	jakarta.apache.org/struts	reines MVC-Framework
Turbine	jakarta.apache.org/turbine	MVC-Umsetzung mit zusätzlichen Diensten
WebWork	opensymphony.com/webwork	einfaches MVC-Framework, nicht auf Web beschränkt
Expresso	www.jcorporate.com	ähnlich Umfangreich wie Turbine, mit Struts-Integration

Tabelle 5.1.: Eine Auswahl von freien Web-Frameworks, die das MVC-Paradigma umsetzen

Das Struts Framework war eines der ersten MVC-Frameworks für Servlets und JSPs. Es hat einen Reifegrad erreicht, der einen Einsatz in kritischen Projekten erlaubt. Es setzt ausschließlich das MVC-Paradigma um. Hierdurch ist es kompakt und übersichtlich geblieben. Struts ist ein sehr populäres Framework, das in der Javawelt sehr häufig eingesetzt wird. Die weite Verbreitung zieht eine große Entwicklergemeine nach sich. Es existiert gute Dokumentation über das Framework und auch die Werkzeugunterstützung, zum Beispiel in Form von Plugins für Entwicklungsumgebungen, ist ausgesprochen gut.

Mit Struts kann der MVC-Ansatz umgesetzt werden. Die Anwendung des Ansatzes wird jedoch nicht sichergestellt. Die JSP können, wie es das MVC vorsieht, ausschließlich zur Darstellung verwendet werden. Es besteht jedoch die Möglichkeit, dass Controller- oder, viel schlimmer, Model-Funktionalität in eine JSP aufgenommen wird. Häufig geschied dies, wenn kleine Änderungen „mal eben schnell“ eingepflegt werden sollen. Dies ist dann zwar

die schnellste Lösung. Sie erhöht jedoch nicht die Wartbarkeit. So wird entsprechend viel Disziplin bei der Entwicklung benötigt.

Turbine ist ein weitreichendes Web-Framework. Es bietet viele bei Webanwendungen benötigte Dienste an. So setzt Turbine nicht nur das MVC-Paradigma zur Entkoppelung von Präsentation, Ablaufsteuerung und Modell/ Datenhaltung um, sondern bringt zusätzlich auch einen eigenen objektrelationalen Mapper (siehe weiter unten im Abschnitt 5.4 auf Seite 63) mit. Die einzelnen Dienste können auch als separate Komponenten und auch ohne die Web-Funktionalität eingesetzt werden. Im Gegensatz zu Struts erzwingt Turbine die strikte Einhaltung der im MVC-Paradigma vorgesehenen Trennung der Komponenten.

Im Unterschied zu den reinen Web-Frameworks, und anders als der Name vermuten lässt, ist WebWork von der Webschicht entkoppelt. Das Framework kann auch in Swing-Anwendungen verwendet werden.

Aufgrund der weiten Verbreitung und des hohen Reifegrades soll hier Struts verwendet werden. Es ist vorstellbar, dies später einmal durch den neuen Javastandard JavaServer Faces (Sun 2004) zu ersetzen. Zurzeit befindet sich der Standard allerdings noch in der Entwicklung.

Persistenzschicht

Die Persistenzschicht umfasst alles, was benötigt wird, um die Daten dauerhaft abzuspeichern. Je nach verwendeter Technik ist das ein Datenbankmanagementsystem oder zusätzlich noch Hilfsmittel, die das Datenbankmanagementsystem mit der objektorientierten Welt verbinden.

Die im Abschnitt 3.4 geforderte Datensicherheit wird teilweise durch das in oder hinter der Persistenzschicht angesiedelte Datenbankmanagementsystem gewährleistet. So wird die Sicherung von Konsistenz, Integrität und Dauerhaftigkeit der Daten durch das Datenbankmanagementsystem sichergestellt. Die persönlichen Daten müssen auch vor unberechtigten Zugriff geschützt werden können. Dieser Aspekt wird hier nicht behandelt, da es den Rahmen dieser Arbeit übersteigen würde und davon ausgegangen wird, dass das verwendete Betriebssystem diese Funktionen (zumindest rudimentär) zur Verfügung stellt. Da es sich hier um eine Webanwendung handelt, muss zusätzlich zur Betriebssystemebene noch die Webschnittstelle abgesichert werden. Dies kann zum Beispiel geschehen, in dem vor den hier verwendeten Webcontainer ein Webserver mit SSL-Verschlüsselung vorgeschaltet wird. Der Webserver wird hierfür als Proxy für den Applikationsserver konfiguriert.

Datenbankmanagementsystem

Die Auswahl der Datenbankmanagementsysteme in Tabelle 5.2 wurde auf die gleiche Weise bestimmt, wie bei den Web Frameworks. Auch hier sollen ausschließlich Produkte betrachtet werden, die so weit verbreitet eingesetzt werden, dass auf eine ausreichende Unterstützung geschlossen werden kann. Der mit Abstand am verbreitetste freie Datenbankserver ist zurzeit MySQL. Erst in einiger Distanz folgt der Datenbankserver PostgreSQL. Später kommt ein breites Feld an freien Datenbankservern. Da bei zwei Produkten aus diesem Feld zu erwarten ist, dass diese in der nächsten Zeit deutlich an Popularität gewinnen dürften, wurden diese aus dem breiten Feld zusätzlich ausgewählt. Die ehemals SAP DB genannte und jetzige MaxDB dürfte beispielsweise von der Bekanntheit von MySQL profitieren. Da Firebird auf den Quellen der verbreiteten Datenbank InterBase aufbaut, kann diese auf die breite InterBase-Unterstützung aufbauen.

Name	Homepage
MySQL	www.mysql.org
PostgreSQL	www.postgresql.org
Firebird	www.firebirdsql.org
SAP DB/ MaxDB	www.sapdb.org

Tabelle 5.2.: Eine Auswahl von freien Datenbankmanagementsystemen

Im Wesentlichen sind alle betrachteten Datenbanken für den Einsatz geeignet. MySQL wird hauptsächlich im Web-Umfeld eingesetzt. Für derartige Aufgaben ist es auch konzipiert worden. MySQL ist eine sehr einfach gehaltene Datenbank, die vor allem auf Geschwindigkeit optimiert ist. Hier steht jedoch primär die Datensicherheit im Vordergrund. Da MySQL hierauf nicht primär ausgerichtet ist, soll dieser Datenbankserver hier nicht verwendet werden.

Firebird glänzt durch seine breite Unterstützung auf Basis der Anwendungsentwicklung. Hauptsächlich aufgrund der Entwicklungswerkzeuge aus dem Hause Borland, dem inzwischen InterBase gehört. Da die Entwicklergemeine noch hauptsächlich auf die proprietären Entwicklungswerkzeuge aus dem Borland-Umfeld setzt, kann auf die breite Unterstützung hier nicht gesetzt werden. MaxDB wird sicherlich in Zukunft stark an Unterstützung hinzugewinnen.

Aufgrund der ausgedehnten Unterstützung und angesichts einiger interessanter zusätzlicher Eigenschaften, soll hier PostgreSQL eingesetzt werden. Neben den klassischen SQL-Funktionen bietet PostgreSQL zusätzlich Funktionen zur Verarbeitung von geometrischen Daten. Hierdurch lassen sich Anfragen formulieren, die etwa alle Punkte innerhalb eines vorgegebenen Bereichs zurückliefern. Es können auch Suchergebnisse nach der Entfernung zu einem vorgegebenen Punkt sortiert werden. Dies könnte für spätere Suchoperationen auf dem persönlichen Wissen interessant sein.

O/R-Mapper

Transparentes Mapping ist mittels O/R-Mapping-Tools realisierbar. Hier gibt es verschiedene Ansätze:

- EJB Persistence
- Java Data Objects (JDO)
- Alternative Mapper

Während EJBs zu komplex sind und zusätzlich einen Application Server erfordern. Ist JDO, als recht junger Standard, noch nicht ganz ausgereift. Vor allem die Abfragesprache ist bei JDO (noch) stark verbesserungswürdig. Da die Java-Standardt-Techniken zur objektrelationalen Abbildung die hier benötigten Anforderungen nicht erfüllen, bleiben noch die alternativen Ansätze übrig.

Name	Homepage
Hibernate	www.hibernate.org
Castor JDO	www.castor.org
Torque	db.apache.org/torque
ObJectRelationalBridge	db.apache.org/ojb

Tabelle 5.3.: Eine Auswahl von freien O/R-Mappern

In Tabelle 5.3 ist eine Auswahl von freien Werkzeugen zur objektrelationalen Abbildung aufgeführt. Hier soll Hibernate eingesetzt werden. Der Mapper arbeitet mit dem PostgreSQL Datenbankserver zusammen, ist umfangreich dokumentiert und stützt sich auf eine große Nutzergemeinde.

Transformation

Zur Verarbeitung von XML-Daten existiert eine Vielzahl von Standardkomponenten, wie etwa XML-Parser. Die zur Transformation der XML-Dokumente benötigten Komponenten können einzeln ausgewählt werden. Das XML-Publishing-Framework Cocoon beinhaltet alle zur Bearbeitung von XML-Daten benötigten Komponenten bereits (Cocoon 2004).

Die XML-Dokumente können zum Beispiel mit OpenOffice (OpenOffice 2004) erstellt werden. OpenOffice ist ein freies multi-plattform Office Programm. OpenOffice benutzt XML als natives Dateiformat.

5.5. Zusammenfassung

Auf der Java-Plattform wurde der Prototyp eines persönlichen Wissensmanagementsystems auf Basis frei verfügbarer Standardkomponenten umgesetzt. Der Quelltext des Prototyps befindet sich auf der beigefügten CD-ROM. Der Prototyp stellt eine erste Näherung dar. Um den Aufwand überschaubar zu halten, wurde vorerst nur eine Webanwendung realisiert. Zur Trennung der Präsentation von den Fachklassen wurde das MVC-Framework Struts verwendet. Als Datenbank kam PostgreSQL zum Einsatz und die Schnittstelle zwischen der Anwendungslogik und der Datenhaltung bildet Hibernate. Innerhalb der Anwendungslogik wird schließlich Cocoon zur Transformation der XML-Daten benutzt.

6. Fazit und Ausblick

Die Zielsetzung lautete, verteiltes Wissen für unterschiedliche Formate nutzbar zu machen. Die so verwertbar gemachten Informationen sollten bei minimalem Aufwand strukturiert abgelegt werden. Dabei sollte der Aufwand beim Gebrauch und die Anschaffungskosten minimal sein. Demgemäß wurde ein System entworfen, welches den geforderten Anforderungen genügt. Zur Benutzung des entwickelten Systems sind keine Vorarbeiten nötig. Es muss kein Ablagesystem erstellt werden, kein Modell der Datenbasis, kein Regelwerk. Auch während der Ablage von neuen Informationen ist der Aufwand minimal, da sich die Einordnung quasi von selbst ergibt. Um die automatische Strukturierung der Informationen zu erreichen, wurde als Startpunkt für den Zugang zum persönlichen Wissen der Terminkalender gewählt. Dieser verknüpft bereits vom Prinzip her viele Informationen. Zur Verwaltung des gesamten persönlichen Wissens wurde der Terminkalender entsprechend erweitert. Des Weiteren wurde der Aspekt berücksichtigt, Informationen jederzeit und an jedem Ort nutzbar zu machen. Der erarbeitete Entwurf wurde mit Standardkomponenten in einer ersten Fassung realisiert. Die Nützlichkeit des Ansatzes und Systems muss sich jedoch erst im täglichen Gebrauch erweisen.

Die Entscheidung für den Einsatz von Java als Plattform wurde hauptsächlich aufgrund der Stabilität der Ausführungsumgebung und der großen Anzahl der frei oder auch kommerziell verfügbaren Komponenten getroffen. Um die Kosten möglichst gering zu halten, wurden ausschließlich die auf dem Markt frei verfügbare Java-Komponenten verwendet. Die Komponenten sind zwar kostenlos Nutzbar, der Aufwand, sie aufeinander abzustimmen ist jedoch nicht zu unterschätzen.

Die vorliegende Arbeit stellt die Basis für zukünftige persönliche Wissensmanagementsysteme dar. Um den gesetzten Rahmen einzuhalten, mussten zwangsläufig verschiedene Themenbereiche ausgeklammert werden. Das System wurde in Form einer Webanwendung umgesetzt. Auf diese Weise kann leicht von überall auf die Daten zugegriffen werden, ohne dass sich um Verfahren zum Abgleich der Daten gekümmert werden musste. Eine zusätzliche Stand-alone-Anwendung wäre sicherlich wünschenswert. Bei Stand-alone-Anwendungen sind ganz andere Möglichkeiten zur Interaktion mit der Anwendung einsetzbar. Einschränkend ist jedoch zu bemerken, dass in diesem Fall der Datenabgleich sehr aufwendig ist. Insbesondere die Möglichkeit der Bearbeitung auf dem Stand-alone-Rechner erhöht den

Aufwand immens. Die Gründe hierfür liegen darin, dass die unterschiedlichen Versionen der Daten zusammengeführt werden müssen.

Oberste Priorität bei der hier umgesetzten zentralen Anwendung hatte die Sicherheit der Daten. Diese Datensicherheit wurde mithilfe einer Datenbank umgesetzt. Bei Stand-alone-Anwendungen besteht die Schwierigkeit, dass die Installation einer Datenbank einen großen Aufwand darstellt. Daraus folgt unter Umständen, dass auf den Stand-alone-Geräten Abstriche bezüglich der Datensicherheit gemacht werden müssten. Alternativ könnten die Daten auf einem zentralen Server mit hohem Sicherheitsstandard abgelegt werden. Dieser Server ist aufgrund der hier verwendeten Webanwendung bereits vorhanden.

Zukünftige Arbeiten können auf der hier erstellten Basis aufbauen. Beispielsweise wären die bereits erwähnten Stand-alone-Applikationen denkbar. Insbesondere im Hinblick auf die Benutzungsoberfläche zur Navigation innerhalb des Wissens könnten unterschiedliche Ansätze erprobt werden.

Vorstellbar ist, dass aufgrund der hohen Verbreitung von Microsoft im Client-Bereich die .NET-Plattform eine so starke Verbreitung findet, dass man die vorliegende Anwendung auch auf dieser Plattform nutzen möchte. Die Verbindung von der bestehenden J2EE-Anwendung zu einer mit dem .NET-Framework umgesetzten Anwendung kann mittels Web Services realisiert werden. Weiterhin können intelligente Verfahren zur Suche ausgetestet werden.

A. Glossar

ACM Association for Computing Machinery; Weltweit größte Vereinigung von Fachleuten und Wissenschaftlern der Informationstechnologie

BLOB engl. Binary Large OBject; Binärobjekte

Content Management Erstellung und Verwaltung von digitalen Inhalten unter besonderer Berücksichtigung von Konsistenz und Erschließbarkeit der Inhalte.

EJB Enterprise Java Beans

GPRS General Packet Radio Service; Übertragungstechnik, die theoretische Bandbreiten von bis zu 171 Kilobit pro Sekunde ermöglicht. GPRS basiert auf der Übermittlung einzelner Datenpakete.

Handheld-PC Computer im Westentaschenformat. Auch PDA (Persönlicher Digitaler Assistent) oder nach dem (ehemaligen) Namen des bekanntesten Herstellers derartiger Computer Palm (palm: engl. Handfläche) genannt.

HSCD High Speed Circuit Switch Data; Übertragungstechnik, die zur Steigerung der Übertragungsgeschwindigkeit mehrere Kanäle bündelt. Dadurch können Datenraten von bis zu 64 Kilobit pro Sekunde erreicht werden.

HTML Hypertext Markup Language; Standardisierte Seitenbeschreibungssprache für Seiten im WWW

J2EE Java 2 Enterprise Edition

Java Beans Konvention für JavaBean-Komponenten

JCP Java Community Process

JDO Java Data Objects

JSP Java Server Pages

Klassifikationssystem Zusammenfassung von Begriffen anhand von klassenbildenen Merkmalen.

Kontext Ein inhaltlicher Sach- und Situationszusammenhang, in dem eine Information steht und aus der sie heraus verstanden werden muss.

Mobile Computing (mobile: engl. beweglich) PDA, Mobiltelefon. . .

MVC Model/ View/ Controller

Ontologie Konzeptuelle Formalisierung von Wissensgebieten, um Teile der Realität auf eine Datenmenge abzubilden. Damit Wissensgebiete für Maschinen verstehbar werden.

OODMBS Objektorientierte Datenbankmanagementsystem

OSS Open Source Software; Software, dessen Quelltexte offen einsehbar ist.

PC Personal Computer

PDA Personal Digital Assistant (siehe Handheld-PC)

PDF Portable Document Format

Pervasive Computing (pervasive: engl. überall vorhanden) Im Vergleich zu Ubiquitous Computing eher pragmatische Herangehensweise, mit dem Ziel die allgegenwärtigen Computer möglichst kurzfristig realisieren zu können.

PIM Personal Information Manager

RDF Resource Description Framework; Standard zur einfachen Erstellung von Beschreibungen.

RDBMS Relationale Datenbankmanagementsystem

RMI Java Remote Method Invocation

Semantic Web Das Wiederfinden und somit die Nutzung von Informationen im WWW soll vereinfacht werden. Vor allem die gezielte Suche nach Informationen soll auf diesem Weg vereinfacht werden. Hierfür sollen die Dokumente des WWW mit maschinenlesbaren Beschreibungen versehen werden (siehe RDF).

Servlet In Java geschriebenes Programmodul, das dazu dient, Anfragen an einen Webserver so zu beantworten.

SSL Secure Sockets Layer; Übertragungsprotokoll mit dem verschlüsselte Kommunikation

SQL Structured Query Language; strukturierte Abfragesprache für Datenbanken

SyncML Synchronization Markup Language; Standard zum Datenabgleich

Ubiquitous Computing (ubique: lat. überall) Von Mark Weiser vom Xerox Forschungszentrum in Palo Alto (PARC) geprägter Begriff. Computer sind in dieser Vision zwar allgegenwärtig. Die Technik tritt aber in den Hintergrund und fungiert nur noch als reines Mittel zum Zweck (Weiser 1991).

UML Unified Modeling Language; standardisierte, graphische Sprache zur Beschreibung objektorientierter Modelle

Wissensmanagement Menschliches Wissen soll informationstechnisch verarbeitbar, auffindbar und transferierbar gemacht werden. Hauptsächlich in Bezug auf Organisationen. Teilweise wird auch in der deutschsprachigen Literatur die englische Bezeichnung „Knowledge Management“ verwendet.

WML Wireless Markup Language; Hypertextsprache für mobile Kleingeräte

WWW World Wide Web, auch WWW oder W3 genannt; graphische Teil des Internets

XML Extensible Markup Language W3C (2004).

XSL Extensible Stylesheet Language

XSLT XSL Transformations

Literaturverzeichnis

Ackoff 1989 ACKOFF, Russell L.: From Data to Wisdom. In: *Journal of Applied Systems Analysis* 16 (1989), Juli, S. 3–9. – ISSN 0308-9541

ACM 1998 ASSOCIATION FOR COMPUTING MACHINERY (Hrsg.): *ACM Computing Classification Systems*. 1998. – URL `http://www.acm.org/class/`. – Zugriffsdatum: 2004-03-05

Ambler 1998 AMBLER, Scott W.: *Mapping Objects to Relational Databases*. Oktober 1998. – URL `http://www.ambysoft.com/mappingObjects.pdf`. – Zugriffsdatum: 2002-01-24

Atkinson 1991 ATKINSON, Malcolm P.: A Vision of Persistent Systems. In: DELOBEL, Claude (Hrsg.) ; KIFER, Michael (Hrsg.) ; MASUNAGA, Yoshifumi (Hrsg.): *Proceedings of Deductive and Object–Oriented Databases (DOOD '91)* Bd. 566. Berlin, Germany : Springer, Dezember 1991, S. 453–459. – ISBN 3-540-55015-1

Atkinson u. a. 1982 ATKINSON, Malcolm P. ; CHISHOLM, K. ; COCKSHOTT, P.: PS-algol: an Algol with a persistent heap. In: *ACM SIGPLAN Notices* 17 (1982), Juli, Nr. 7, S. 24–31. – ISSN 0362-1340

Atkinson und Morrison 1995 ATKINSON, Malcolm P. ; MORRISON, Ronald: Orthogonally persistent object systems. In: *The VLDB Journal - The International Journal on Very Large Data Bases* 4 (1995), Nr. 3, S. 319–402. – ISSN 1066-8888

Banavar und Bernstein 2002 BANAVAR, Guruduth ; BERNSTEIN, Abraham: Software infrastructure and design challenges for ubiquitous computing applications. In: *Communications of the ACM* 45 (2002), Nr. 12, S. 92–96. – ISSN 0001-0782

Beck 1999 BECK, Kent: *Extreme Programming Explained: Embracing Change*. Addison-Wesley, 1999

Berners-Lee 1989 BERNERS-LEE, Tim: *Information Management: A Proposal*. 1989. – URL `http://www.w3.org/History/1989/proposal.html`. – Zugriffsdatum: 2004-03-05

Beyer u. a. 2001 BEYER, Lothar ; OERTEL, Britta ; STEINMÜLLER, Karlheinz: *WerkstattBerichte*. Bd. 49: *Entwicklung und zukünftige Bedeutung mobiler Multimediadienste*. 1. Auflage. Berlin : IZT - Institut für Zukunftsstudien und Technologiebewertung GmbH, 2001. – ISBN 3-929-173-49-2

Brown und Whitenack 1996 BROWN, Kyle ; WHITENACK, Bruce G.: Crossing Chasms: A Pattern Language for Object-RDBMS Integration. In: VLISSIDES, John M. (Hrsg.) ; COPLIEN, James O. (Hrsg.) ; KERTH, Norman L. (Hrsg.): *Pattern Languages of Program Design 2*. Addison Wesley, 1996, S. 227–238. – URL `ftp://members.aol.com/kgb1001001/Chasms/chasms.pdf`. – Zugriffsdatum: 2004-03-05

Buschmann u. a. 1996 BUSCHMANN, Frank ; MEUNIER, Regine ; ROHNERT, Hans ; SOMMERLAD, Peter ; STAL, Michael: *A System of Patterns: Pattern-Oriented Software Architecture*. New York , NY , USA : John Wiley & Sons, 1996. – ISBN 0-471-95869-7

Bush 1945 BUSH, Vannevar: As we may think. In: *Atlantic Monthly* 176 (1945), Juli, Nr. 1, S. 101–108. – URL `http://www.theatlantic.com/unbound/flashbks/computer/bushf.htm`. – Zugriffsdatum: 2004-03-05

Catford 1969 CATFORD, John: Learning a Language in the Field: Problems of Linguistic Relativity. In: *Modern Language Journal* 53 (1969), Nr. 5

Churchman 1971 CHURCHMAN, C. W.: *The Design of Inquiring Systems*. Basic Books, 1971

Cleveland 1982 CLEVELAND, Harlan: Information As a Resource. In: *The Futurist* (1982), Dezember, S. 34–39. – ISSN 0016-3317

Cocoon 2004 APACHE SOFTWARE FOUNDATION (Hrsg.): *The Apache Cocoon Project*. 2004. – URL `http://cocoon.apache.org/`. – Zugriffsdatum: 2004-05-25

Collins und Quillian 1969 COLLINS, A. M. ; QUILLIAN, M. R.: Retrieval Time from Semantic Memory. In: *Journal of Verbal Learning and Verbal Behavior* 8 (1969), S. 240–247

Date 2000 DATE, C. J.: *The Database Relational Model: A Retrospective Review and Analysis*. Addison-Wesley Longman Publishing Co., Inc., Mai 2000. – ISBN 0201612941

Davenport u. a. 1997 DAVENPORT, Thomas H. ; LONG, David W. D. ; BEERS, Michael C.: Building Successful Knowledge Management Projects. (1997), Januar

Davenport und Prusak 1997 DAVENPORT, Thomas H. ; PRUSAK, Laurence: *Working Knowledge: How Organizations manage what they know*. Paperback, May 2000, ISBN: 1578513014. Cambridge, MA, USA : Harvard Business School Press, Dezember 1997. – ISBN 0875846556

Davis 2002 DAVIS, Gordon B.: Anytime/anyplace computing and the future of knowledge work. In: *Communications of the ACM* 45 (2002), Dezember, Nr. 12, S. 67–73. – ISSN 0001-0782

DRSC 2002 DEUTSCHES RECHNUNGSLEGUNGS STANDARDS COMMITTEE E.V.: DRS 12 Immaterielle Vermögenswerte des Anlagevermögens. In: *Bundesanzeiger* (2002), Oktober

Drucker 1993 DRUCKER, Peter F.: *Post-capitalist society*. Paperback, 1998, ISBN: 0-7506-2025-0. Oxford : Butterworth Heinemann, April 1993. – ISBN 0-7506-0921-4

Englert 2001 ENGLERT, Sylvia: Die überforderte Ich-AG. In: *changeX* (2001), Dezember. – URL `http://www.changex.de/d_a00433.html`. – Zugriffsdatum: 2004-03-05

eXist 2004 MEIER, Wolfgang (Hrsg.): *eXist: Open Source XML Database*. 2004. – URL `http://www.exist-db.org/`. – Zugriffsdatum: 2004-05-19

freshmeat 2004 NETWORK, Open Source D. (Hrsg.): *freshmeat*. 2004. – URL `http://freshmeat.net/about/`. – Zugriffsdatum: 2004-06-01

Gamma u. a. 1994 GAMMA, Erich ; HELM, Richard ; JOHNSON, Ralph ; VLISSIDES, John M.: *Design Patterns: Elements of Reusable Object-Oriented Software*. 23. print, 2002, ISBN: 0-201-63361-2. Reading, MA, USA : Addison-Wesley, Oktober 1994. – ISBN 0-201-63361-2

Gemmell u. a. 2002 GEMMELL, Jim ; BELL, Gordon ; LUEDER, Roger ; DRUCKER, Steven ; WONG, Curtis: MyLifeBits: fulfilling the Memex vision. In: *Proceedings of the tenth ACM international conference on Multimedia*, ACM Press, 2002, S. 235–238. – URL `http://research.microsoft.com/~jgemmell/pubs/MyLifeBitsMM02.pdf`. – Zugriffsdatum: 2004-03-05. – ISBN 1-58113-620-X

Gruber 1993 GRUBER, T. R.: A translation approach to portable ontology specifications. In: *Knowledge Acquisition* 5 (1993), Nr. 2, S. 199–220

Hansmann u. a. 2001 HANSMANN, Uwe ; NICKLOUS, Scott ; MERK, Lothar ; STOBER, Thomas: *Pervasive Computing Handbook*. 1. Auflage. Heidelberg : Springer Verlag, 2001. – ISBN 3-540-67122-6

de Icaza 2004 ICAZA, Miguel de: *Mono project*. 2004. – URL `http://www.go-mono.com/`. – Zugriffsdatum: 2004-05-18

infoAsset 2004 INFOASSET AG (Hrsg.): *Der infoAsset Broker: Software für das Wissensmanagement*. 2004. – URL `http://www.infoasset.de/contents/products/index.htm`. – Zugriffsdatum: 2004-05-15

Kargar u. a. 1999 KARGAR, David ; ADAR, Eytan ; STEIN, Lynn A.: Haystack: per-user information environments. In: *Proceedings of the eighth international conference on Information and knowledge management*, ACM Press, 1999, S. 413–422. – ISBN 1-58113-146-1

Karger 2003 KARGER, David R.: *Haystack*. 2003. – URL `http://haystack.lcs.mit.edu/`. – Zugriffsdatum: 2004-05-15

KBSt 2003 KBST (Hrsg.): *KBSt-Schriftenreihe*. Bd. 59: *SAGA - Standards und Architekturen für E-Government-Anwendungen*. Berlin : Bundesministerium des Innern, Dezember 2003. – URL `http://www.kbst.bund.de/Anlage304273/pdf_datei.pdf`. – Zugriffsdatum: 2004-05-13. ISSN 0179-7263

Kubicek 1985 KUBICEK, H.: *Die sogenannte Informationsgesellschaft*. S. 76–109. In: ALTVATER, Elmar (Hrsg.) ; BAETHGE, Martin (Hrsg.) ; BÄCKER, Gerhard (Hrsg.): *Arbeit 2000. Über die Zukunft der Arbeitsgesellschaft.* Hamburg : VSA-Vlg., 1985. – ISBN 3-87975-314-8

Kuhlen 1991 KUHLEN, Rainer: *Hypertext – Ein nicht-lineares Medium zwischen Buch und Wissensbank*. Berlin : Springer, 1991. – ISBN 3540535667

Linton 1975 LINTON, Marigold: Memory for real-world events. In: NORMAN, Donald A. (Hrsg.): *Explorations in Cognition*. San Francisco, CA, USA : W H Freeman & Co., September 1975, S. 376–404

Lorenz 1993 LORENZ, Mark: *Object-Oriented Software Development*. Englewood Cliffs : Prentice Hall, 1993. – ISBN 0-13-726928-5

Lyytinen und Yoo 2002 LYYTINEN, Kalle ; YOO, Youngjin: Introduction. In: *Communications of the ACM* 45 (2002), Dezember, Nr. 12, S. 62–65. – ISSN 0001-0782

Microsoft 2004 MICROSOFT CORPORATION (Hrsg.): *MyLifeBits Project*. 2004. – URL `http://research.microsoft.com/barc/MediaPresence/MyLifeBits.aspx`. – Zugriffsdatum: 2004-05-15

Miller 2003 MILLER, Gerry: The Web services debate: .NET vs. J2EE. In: *Commun. ACM* 46 (2003), Nr. 6, S. 64–67. – ISSN 0001-0782

Netcraft 2003 NETCRAFT LTD (Hrsg.): *November 2003 Web Server Survey*. November 2003. – URL `http://news.netcraft.com/archives/2003/11/03/november_2003_web_server_survey.html`. – Zugriffsdatum: 2004-05-21

OMA 2004 OPEN MOBILE ALLIANCE (Hrsg.): *SyncML*. 2004. – URL `http://www.syncml.org/`. – Zugriffsdatum: 2004-05-18

OpenOffice 2004 OPENOFFICE.ORG ORGANIZATION (Hrsg.): *OpenOffice*. 2004. – URL `http://www.openoffice.org/`. – Zugriffsdatum: 2004-05-25

Parnas 1978 PARNAS, David L.: Designing software for ease of extension and contraction. In: *Proceedings of the 3rd international conference on Software engineering*, IEEE Press, 1978, S. 264–277. – ISBN none

Parnas 1972 PARNAS, David L.: On the criteria to be used in decomposing systems into modules. In: *Communications of the ACM* 15 (1972), Dezember, Nr. 12, S. 1053–1058. – ISSN 0001-0782

Polanyi 1966 POLANYI, Michael: *The Tacit Dimension*. Nachdruck Peter Smith, Gloucester, Massachusetts, 1983, ISBN: 0-844-65999-1. Garden City, N.Y., USA : Doubleday & Co., 1966. – ISBN 0-385-06988-X

Preece 1981 PREECE, Scott E.: *A spreading activation network model for information retrieval*, University of Illinois at Urbana-Champaign, Dissertation, 1981

Reenskaug 1979a REENSKAUG, Trygve: *MODELS - VIEWS - CONTROLLERS*. Dezember 1979. – URL `http://heim.ifi.uio.no/~trygver/1979/mvc-2/1979-12-MVC.pdf`. – Zugriffsdatum: 2004-02-02

Reenskaug 1979b REENSKAUG, Trygve: *THING-MODEL-VIEW-EDITOR*. Mai 1979. – URL `http://heim.ifi.uio.no/~trygver/1979/mvc-1/1979-05-MVC.pdf`. – Zugriffsdatum: 2004-02-02

Reenskaug u. a. 1995 REENSKAUG, Trygve ; WOLD, P. ; LEHNE, O. A.: *Working with Objects: The OORam Software Engineering Method*. Prentice-Hall, 1995

Ross u. a. 1975 ROSS, D. T. ; GOODENOUGH, J. B. ; IRVINE, C. A.: Software Engineering: Process, Principles, and Goals. In: *IEEE Computer* 8 (1975), Mai, Nr. 5, S. 17–27. – ISSN 0018-9162

Schacter u. a. 1984 SCHACTER, D.L. ; HARBLUK, J.L. ; MCLACHLAN, D.R.: Retrieval without recollection: An experimental analysis of source amnesia. In: *Journal of Verbal Learning and Verbal Behavior* 23 (1984), Nr. 5, S. 593–611

Schuster und Wilhelm 2000 SCHUSTER, Erwin ; WILHELM, Stephan: Content Management. In: *Informatik-Spektrum* 23 (2000), Dezember, Nr. 6, S. 373–375. – ISSN 0170-6012

Simon 1962 SIMON, Herbert A.: The Architecture of Complexity: Hierarchic Systems. In: *Proceedings of the American Philosophical Society* 106 (1962), Dezember, S. 467–482

Sun 2002 SUN MICROSYSTEMS, INC. (Hrsg.): *J2EE Patterns: Model-View-Controller*. 2002. – URL `http://java.sun.com/blueprints/patterns/MVC.html`. – Zugriffsdatum: 2004-05-26

Sun 2004 SUN MICROSYSTEMS, INC. (Hrsg.): *JavaServer Faces*. 2004. – URL `http://java.sun.com/j2ee/javaserverfaces/`. – Zugriffsdatum: 2004-04-29

Sveiby 1997 SVEIBY, Karl-Erik: *The New Organizational Wealth: Managing and Measuring Knowledge-Based Assets*. Nachdr. San Fransisco, CA, USA : Berrett-Koehler, April 1997. – ISBN 1-57675-014-0

Visser 2004 VISSER, Ubbo: *Publications*. 2004. – URL `http://www.tzi.de/~visser/publications.html?year=ally&topic=sw`. – Zugriffsdatum: 2004-05-15

W3C 1999 WORLD WIDE WEB CONSORTIUM (Hrsg.): *XSL Transformations (XSLT)*. 1999. – URL `http://www.w3.org/TR/xslt`. – Zugriffsdatum: 2004-04-30

W3C 2001 WORLD WIDE WEB CONSORTIUM (Hrsg.): *Semantic Web*. 2001. – URL `http://www.w3.org/2001/sw/`. – Zugriffsdatum: 2004-04-30

W3C 2004 WORLD WIDE WEB CONSORTIUM (Hrsg.): *Extensible Markup Language (XML) 1.0 (Third Edition)*. 2004. – URL `http://www.w3.org/TR/REC-xml/`. – Zugriffsdatum: 2004-04-30

Weiser 1991 WEISER, Mark: The computer for the 21st century. In: *Scientific American* 265 (1991), September, Nr. 3, S. 94–104. – URL `http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html`. – Zugriffsdatum: 2004-03-05

Williams 2003 WILLIAMS, Joseph: The Web services debate: J2EE vs. .NET. In: *Commun. ACM* 46 (2003), Nr. 6, S. 58–63. – URL `http://computer.org/proceedings/saint/1872/18720009.pdf`. – ISSN 0001-0782

Wirth 1971 WIRTH, Niklaus: Program development by stepwise refinement. In: *Communications of the ACM* 14 (1971), April, Nr. 4, S. 221–227. – ISSN 0001-0782

Xindice 2004 APACHE SOFTWARE FOUNDATION (Hrsg.): *Apache Xindice*. 2004. – URL `http://xml.apache.org/xindice/`. – Zugriffsdatum: 2004-05-19

XMLDB 2004 XML:DB INITIATIVE (Hrsg.): *XML:DB*. 2004. – URL `http://www.xmldb.org/`. – Zugriffsdatum: 2004-05-15

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §24(5) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 4. Juni 2004	
Ort, Datum	Unterschrift